# SEGA® COMPUTER

SEPTEMBER ISSUE 1984

REVIEWS

Sega Hard Keyboard
Sega Colour Monitor
Latest Software Releases

FEATURES

Beginning Programing
Unexplained Commands (2)
Machine Code Programing
Step By Step
Sprites Made Easy

GAMES PROGRAMS

Candy Kid
Gunslinger
And Many
Graphics Classics

The Official Sega User Club Magazine

# WELCOME

## to the SEGA USERS CLUB

Producing a magazine such as the Sega User magazine has been described by many people as being similar to rowing single handed across the Atlantic, only without getting wet; you get just as many blisters on your fingers, and one wrong move and you are liable to get eaten by the sharks.

The effort of putting it all together in both cases is quite considerable, but the rewards at the end of the line make it all worthwhile. In our case rather than the flags and banners of a shore side welcome, our first issue has been greeted with the cheers of the many Sega users who had been waiting with baited breath for more written material on their favourite home computer the Sega SC3000.

Fortunately for us none of the sharks reaching the first issue had very sharp teeth, as when we dipped our toes in the water more than once with printing errors, none of them have so far been bitten off. However, I feel sure that future issues will provide you with far less opportunity to practice your origami on the inserted sheets carrying corrections.

In this second issue we will attempt to enlarge on the guidelines set out in the first, and in many cases, references are made to certain sections of Issue 1, and therefore I strongly recommend you keep all user magazines in a safe place, in order that you can fully understand our future issues.

This month we are featuring same of the lesser known, but extremely versatile aspects of the Sega. In particular, the music cartridge. To date this seems to have been filed along with so many other excellent products in the "too difficult" category, and consequently has so far gone largely unnoticed, by both dealers, and Sega users alike.

In actual fact, when you scratch beneath the surface, you find a real gem of a package, which is very well documented, and once employed, very easy to comprehend, extremely versatile, entertaining, and educational. Anyone who is remotely musically inclined is sure to be interested and impressed with our article on this cartridge, and its operation.

For the programmers section we have more to increase your enlightenment from B. Brown, and plenty of useful hints and routines in our letters section. A new regular columnist is Colin Smith, who will be introducing a good many people to machine code programming as painlessly as possible over the next few issues, and I take this opportunity to welcome him into our ranks.

One area which undoubtedly needs a good shot in the arm is the local area meetings page. This information did not exactly come pouring in to our office, in fact, it would not even amount to a damp spot on the carpet. I know you're all out there, so let us help you get together. Anyone wishing to contact similarly keen Sega users can have contact made through the magazine, and we are keen to help in whatever way we can.

Finally, a short story which I would like to share with you all which happened recently to our sales technician, when he went to the aid of an acquaintance who had recently purchased a Sega computer, and was finding tuning it to the television a little confusing.

For some 15 minutes he struggled to achieve a good balance between contrast, colour intensity, brightness, even the volume, which seemed to be constantly changing. Each time he increased the colour, as soon as he changed the contrast the colour would fade, in fact none of the controls seemed to affect the area they were supposed to, and when the channels started to change for no good reason, he decided to call it a day and suggest the owner take the TV in for repair as it was faulty.

However, as soon as he turned to break the bad news, the real reason for the TV's odd behaviour became clear, as seated in the armchair apparently supervising the whole operation was the owner, armed with the remote control television tuner.

If you know of any similar stories and you want to share them, let me know, a good chuckle never hurt anyone. I hope you enjoy the magazine.

Yours faithfully,
P. Kenyon
GRANDSTAND LEISURE LTD

**LOCAL SEGA USERS CONTACTS**

**ROTORUA**
Rotorua Sega Users Club
C/- 61 Devon Street
ROTORUA

**TOKOROA**
Tokoroa Sega Users Group
C/- 1 Pio Pio Place
TOKOROA
Contact Geoff Phone Number 67105
Tokoroa

**TARANAKI**
South Taranaki Microcomputer Society
D. M. Beale
7A Clive Street
HAWERA

**NAPIER**
Napier Sega Users Club
Sec E. P. Lins
41 Higgins Street
NAPIER

The above are contact names and addresses for Sega Users Clubs. If you wish to have your club advertised write to Sega Users Club P.O. Box 2353, Auckland

# CREATING YOUR OWN PROGRAMS

**By Phil Kenyon**

**Grandstand Leisure**

In issue one the first seven commands covered dealt with only Text, and we therefore, were not required at any stage to call on Sega's other screen, which is the graphic screen otherwise known as the drawing screen.

To look at the drawing screen whilst it is blank simply hold down the shift key and push the break key to return to the text screen. Repeat the operation. You can do this at any stage and should you break into a program which was running on the graphic screen, you can swap back and forth to see what was happening on that screen at the time the program was interrupted.

There are several differences in the text and graphic screens. Firstly the way in which you position information on the screen. Both screens have grid reference points, which are X and Y co-ordinates starting at the top left hand corner of the screen, going 1st across from left to right which is called X, then down the screen from top to bottom or the Y co-ordinate.

With the text screen you have co-ordinates which range from Ø to 39 in the X direction and from Ø to 23 in the Y direction, to position your information, however, there are different types of information which can be displayed, one type of information is a character from the keyboard, the other type is a user definable graphic, and these can be positioned in two different ways, using either the "VPOKE" command which uses a Hexideciaml address in the computers own ROM memory, and therefore, requires an understanding of the computers memory map, or by using the "cursor" and "print" command, which is what we will cover in this issue as it is adequate for most requirements and extremely easy to use at this stage.

When using the cursor command there is a slight limitation using the text screen, and that is when you are positioning characters you have only 38 possible X locations, therefore, to position a letter text the resolution is 38 x 24, therefore, to position a letter or character on screen you would use the following command:

```
CURSOR 18,12:PRINT X
```

now try

```
CURSOR 38,24:PRINT X
```

The reason you get an error is the computer counts from Ø to 37, and Ø to 23, so even though you have a 38 x 24 resolution you cannot exceed 37 or 23 in your co-ordinates.

On the other hand with the graphic screen, rather than addressing an area consisting of a group of dots which are required to display each character on the text screen, you can address one individual dot anywhere on the screen.

In this case the grid consists of 256 dots (or pixels as they are known) in the X direction, and 192 pixels in the Y direction, where 0,0 would be the top left hand corner and 255,191 would be the bottom right.

To position our X in this screen we must first "call" the graphic screen in a program, and devise new co-ordinates accordingly for the centre of the screen resolution:

```
10 SCREEN 2,2:CLS
20 CURSOR 125,95:PRINT "X"
30 GOTO 30
```

Here LINE 10 is the command which tells the computer we want to use the graphic screen, CLS clears the screen.

LINE 20 gives co-ordinates corresponding to the centre of the screen where we wish to put "X", if no other command were given at this stage when we "run" the program, you will see the graphic screen appear, the cross will flash on then rapidly return us to our program on the text screen.

LINE 30 Creates a loop which keeps the information on screen, and can only be interrupted with the Break Key or Reset.

By using break, this will bring you back to the text screen with your program displayed. Reset will result in the computers title screen, reappearing, your program is not lost, you will need to enter "List" to have it displayed.

Now add another "Print" statement after the cursor line:

```
10 SCREEN 2,2:CLS
20 CURSOR 125,95:PRINT "X"
25 PRINT "NEW LINE"
30 GOTO 30
```

The computer will now take the cursor location as its starting point and continue down the screen with all future print commands on lines below the line specified by the cursor command. Anything which is required above the line will require a new cursor location before the print command. This can be used in the following way to display information next to it's description, where the information may be changing but the description remains the same eg:

```
10 SCREEN 2,2
20 FOR A = 1 to 6
30 CLS:CURSOR 125,95:PRINT
   "SCORE";A
40 FOR T = 1 to 300
50 NEXT T,A
60 GOTO 20
```

Where each time the computer counts from 1 to 300 it increases the value of A from 1 to 6 and it is displayed as a score.

There is another difference here between the text and graphic screens. Try moving CLS from the beginning of Line 30, to the end of Line 10 and run the program again. See how the numbers overlap? If you change the program to run in the text screen ie:

```
LINE 10 SCREEN 1,1:CLS
LINE 30 CURSOR 18,12:PRINT
"SCORE";A
```

Here the previous value is erased before it is replaced.

With the graphic screen this must be done with a programming command. Either using CLS which will clear the whole screen, or by printing " " (NOTHING) at the cursor location which needs to be cleared, before the new value is replaced.

## PSET PRESET

Sega has a graphics chip which has many commands which enable you to perform tasks which on most older computers would take many hours of complicated programming and mathematical equations. Particularly when designing pictures on the graphics screen. It also has the ability to enable you to pick out any one of the 50,000 dots which make up the graphic resolution, and have that dot lit in any one of Segas 16 colours.

This command is called PSET

Lets create a dot on screen:

```
10 SCREEN 2,2:CLS
20 PSET (150,100),8
30 GOTO 30
```

By changing any of the co-ordinates for a variable we can produce more than one dot. The colour of the dot is derived from the number after the comma, the table showing the number codes for all 16 colours is:

| | |
|---|---|
| Ø Transparent | 8 Red |
| 1 Black | 9 Light red |
| 2 Green | 10 Deep yellow |
| 3 Light Green | 11 Light yellow |
| 4 Dark blue | 12 Dark green |
| 5 Light blue | 13 Magenta |
| 6 Dark red | 14 Gray |
| 7 Cyan | 15 White |

The co-ordinates must now be enclosed in brackets so as not to become confused with any other values:

Try the following:

```
10 SCREEN 2,2:CLS
20 X = 10
30 PSET (X,100),8
40 X = X + 10
50 IF X = 250 THEN 20
60 GOTO 30
```

If we wish to erase the dots as they travel across the screen, this is achieved using preset. This command is used in exactly the same way, except that it turn the dot off.

The time delay loop is required to slow the process down as otherwise the pixel is switched on and off so quickly the human eye would not be able to see what was happening.

```
10 SCREEN 2,2:CLS
20 X = 10
30 PSET (X,100),8
40 FOR T = 1 to 10:NEXT T
50 PRESET (X,100)
60 X = X + 10
70 IF X = 250 THEN 20
80 GOTO 30
```

By changing both co-ordinates it is possible to plot curves for charts or graphs. Try the following program:

```
10 SCREEN 2,2:CLS
20 FOR C=1 TO 14
30 FOR X=0 TO 255
40 PSET(X,SIN(X/10)*C*6+95),C
50 NEXT X
60 NEXT C
70 GOTO 70
```

LINE 10: Initialises the graphic screen and clears it.
LINE 20: Steps variable C for 1 to 14 is used to change the colour and size of the sin wave.
LINE 30: Starts the for next load controlling variable X, which is used to step the dots across the screen.
LINE 40: Draws the dots with a PSET statement as X is increased plus line is drawn across the screen.

## LINE, CIRCLE, PAINT

Sega's advanced basic, gives you certain commands which are so advanced, the majority of older computers simply cannot perform the tasks which are one line programs with Sega, among these commands are Line, circle and paint.

The method for using the commands are much the same as with PSET, in that you are plotting co-ordinates on a screen matrix. The main difference with "line" is that you have to plot two co-ordinates, one to tell the computer where the line starts, the other to tell it where the line must end.

```
10 SCREEN 2,2:CLS
20 LINE (50,50)-(150,150),5
30 GOTO 30
```

Where the line starts at 50 pixels across on the X co-ordinate, by 50 pixels down on the Y co-ordinate, and finishes 150 across by 150 down.

With diagonal lines such as the one above it is possible to convert them into a square by adding two additonal characters to line 20, a comma, which tells the computer something else is to be read, and the letter B for BOUNDARY.

```
20 LINE (50,50)-(150,150),5,B
```

To make the square one solid colour we need add only one character, the letter F for FILL.

```
20 LINE (50,50)-(150,150),5,BF
```

This has great applications when designing backgrounds for games, or bar charts for financial programs.

For additional speed of programming it is also possible to link a line to the last co-ordinate of the previous line. By omiting to give a first co-ordinate, the computer will automatically take the last one given and substitute that.

```
20 LINE (50,50)-(150,150),5
30 LINE - (50,175),8
```

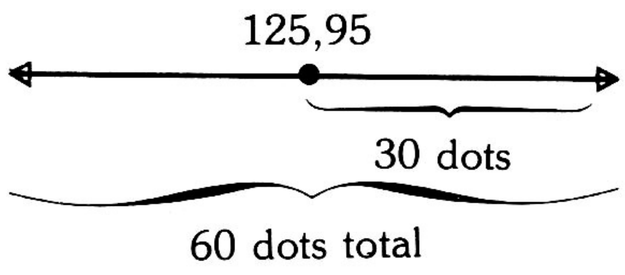
Once again the plotting of charts becomes much more simple and less time consuming. The program dissection shows some clear uses of the line statement for games programming.

## CIRCLE

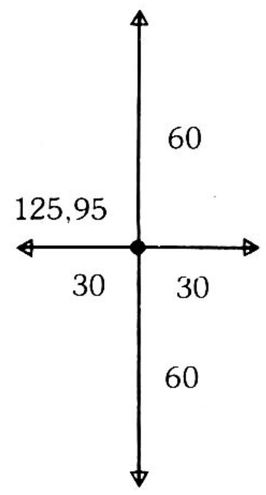
The program later in the magazine DOLL shows just how versatile circle can be, and remarkably easy to use. The format is back to one initial co-ordinate to position the circle, and this is taken as being the centre of our circle.

```
10 SCREEN 2,2:CLS
20 CIRCLE (125,95),30,8,2,0,.75
30 GOTO 30
```

Here in line 20 the centre point of the circle is roughly in the middle of the screen the number 30 after the first comma is the radius and it tells the computer to count out 30 pixels from the centre of the circle and say that is the width of the circle there for the total diameter would be 60 dots ie:



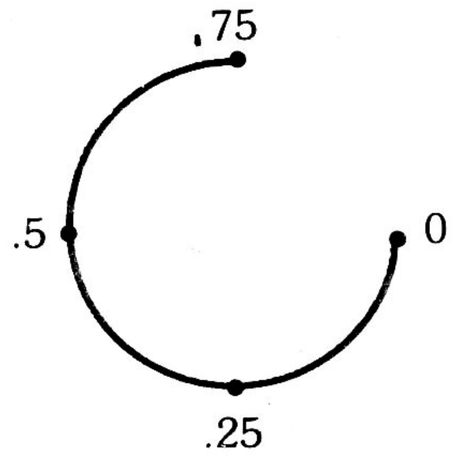

The next number ,8, relates to the colour, which is red. This is followed by the ratio of height to width. In this case we have chosen 2. The computer will now draw the circle twice as high as it is wide.




If the figure entered here was less than one, ie .5, your circle would be half as high as it was wide and appear very squashed.

The next two figures are the start and finish points of the circle.

0 starts the circle off immediately to the right of centre. 1 finishes it in the same place making a full circle. In our example the circle finishes at .75 we therefore have only ¾ of a circle.




You can change the start and finish points to whatever numbers between 0 and 1 you require depending on how much of the circle is needed.

As with the line statement you can put a boundary on the circle, and fill in with colour, add them to your program one at a time to see the effect of B, and BF after the .75 (Don't forget the comma).

Now try putting another different coloured circle on top of the original circle once it is filled with colour.

```
LINE 25 CIRCLE
(125,95),15,4,1,0,1,BF
```

Quite a mess isn't it!

This is where BCIRCLE comes in. The way to create a circle within a circle is to erase the circle. Change line

```
25 to BCIRCLE (125,95),15,,,,,BF.
```

By not inserting any values between commas the computer sets it's own which will be colour 0, tranparent, ratio 1, start 0, end 1.

BLINE works in exactly the same way to erase lines and boxes from the graphic screen.

This blocking effect which is caused by large areas filled with colour overlapping is not a fault with the computer nor is it caused by any programming error, there are certain rules which must be applied when working with graphics which are laid down by the limitations of the video display chip in the computer.

Generally with horizontal lines you need leave only one pixel spece betwen one filled area and another. With vertical lines and horizontal lines you must leave 8 pixels space to avoid one colour "Bleeding" into another colour. This will result in blocks being formed with circular and horizontal lines. This merging of colours does not occur when pixels are addressed individually as patterns or with PSET.

## PAINT

This command enables you to paint in specific areas, which are enclosed by lines, (once again cleverly demonstrated in the doll program). Try the following:

```
10 SCREEN 2,2:CLS
20 LINE (50,50)-(100,50),1
30 LINE - (75,100),1
40 LINE - (50,50),1
50 PAINT (75,75),5
60 GOTO 60
```

Here the three lines form a triangle which on line 50 is painted in, in blue. Any co-ordinate which is inside the area enclosed by the triangle would produce the required result.

Try changing the co-ordinates to one which is outside the triangle.

```
50 PAINT (200,140),5
```

All areas which are not completely enclosed by lines are painted in. That goes for circles too, the area within the circle would be ignored.

So you can see it is very easy really to produce some quite complex and colourful background scenes, with just five or six basic commands, you can produce some very intricate designs.

FOR, TO, NEXT, STEP

We have been using the above commands so far mainly to create a time delay in the running of a program, in other words to make the computer count from X to X, depending on how much it has to count, will determine how long the delay lasts. This command will also help you to control values and even set limits to movement in your programs.

The way the command works is to load a sequence of numbers into a labelled variable ie:

```
FOR A = 1 to 10:NEXT A
```

tells the computer to create an address called A, then put the numbers from 1 up to 10 in that box. Originally A will be equal to 1, until it reaches the part of the program which says NEXT A. The program will then return to the point where A was established and change it to 2, which is the next number in sequence, and the program continues from that point onward, only when all 10 A's are used up will the program continue on past the NEXT A statement.

By using step, you can increase the size of the steps the program will take or increase the speed at which something appears to travel across the screen.

```
10 SCREEN 2,2:CLS
20 FOR A = 10 to 200 STEP 2
30 LINE (A,20) - (A,150),8
40 FOR T = 1 to 10:NEXT T
50 BLINE (A,20) - (A,150)
60 NEXT A
70 PRINT "FINISH"
```

By increasing the size of the step of reducing the amount of the delay the whole program will appear to move more quickly.

Up to 16 for next statements can be "NESTED" as above where we have used two, however it is important to remember the sequence which they are established and to have the NEXT statements the reverse of the FOR, eg:

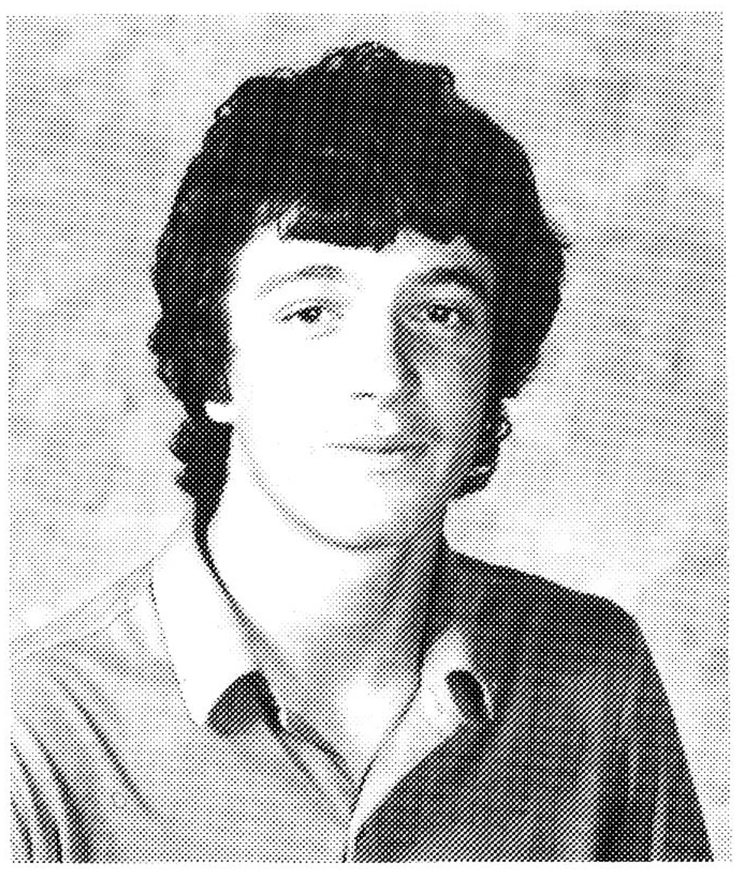
```
10 FOR A = 1 to 10
20 FOR B = 1 to 20 STEP 2
30 FOR C = A to B
40 PRINT C:NEXT C,B,A
```

Should line 40 read, NEXT A, B, C, the program would crash as it would have to go back to the line which reads FOR B = without having encountered a NEXT B statement. This causes a FOR without NEXT ERROR.

---

# PROGRAM REVIEW

## TEACH YOURSELF BASIC GAMES PROGRAMMING



**By Michael Howard**

Release Early September

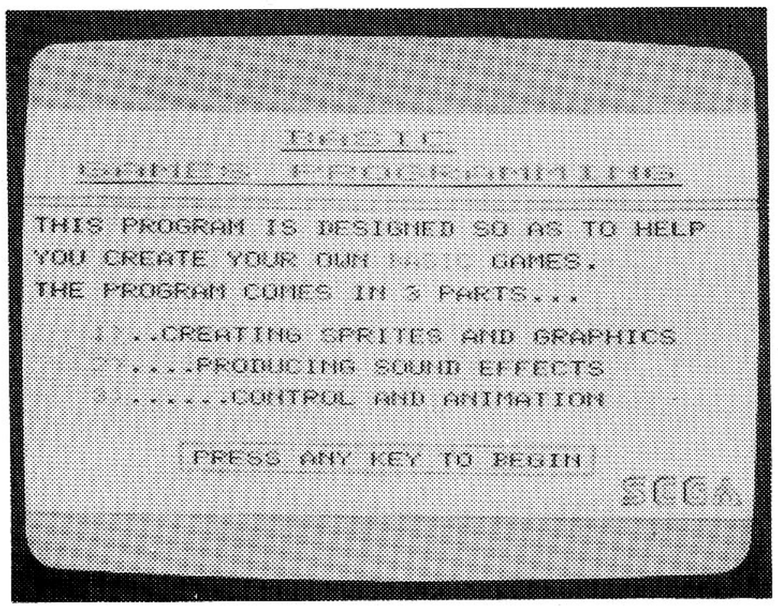
The Teach Yourself package works on the same theory as a teach yourself guitar, or organ book. Rather than buying a book which shows you that if you put your left hand here, and your right hand there, it makes this sound, you would buy a book which taught you how to play Paul McCartney songs, or Simon and Garfunkel — the type of things you enjoy playing. At the same time of course, you are still learning how to play a musical instrument.

With this in mind, we have introduced a program and book, which will show you how to produce something which is enjoyable, has a practical end result, and will increase your knowledge and ability in all areas.

This long awaited computer assisted instruction program, has been painstakingly put together over several months. It is designed to give the clearest possible understanding of the rudimentary requirements of almost every type of graphic computer game, beginning by teaching you the required elementary mathematic knowledge, teaching you the various methods the computer has to assist you in this area, and showing you the easiest possible way of creating your own shapes, on screen, and incorporating sound effects and music to accompany them.

You are then very clearly and logically taught how to position these shapes and control their movement around the screen, both using the joysticks or from the keyboard. You will learn how to program the computer to check whether these objects have collided with anything, and the logic behind making one object seek and chase another object.

Finally, your first game is created with clear precise line by line instruction to show you how your new found knowledge and talents can be put to best


THIS PROGRAM IS DESIGNED SO AS TO HELP
YOU CREATE YOUR OWN GAMES.
THE PROGRAM COMES IN 3 PARTS...
..CREATING SPRITES AND GRAPHICS
...PRODUCING SOUND EFFECTS
.....CONTROL AND ANIMATION
PRESS ANY KEY TO BEGIN
SEGA

use, giving you the understanding and inspiration to carry on and produce your own particular style of computer game.

Not only Basic Commands are covered in the extensive book, which is supplied with the program, but also machine language programming is introduced. The commands and formula to position information on screen without the computer having to first interpret the basic language, enables operators to produce a program which will run more quickly, and have movement which is more smooth across the screen. In the T.Y.B.G.P. manual, these instructions are very clearly explained, and great pains have been taken to ensure any possible confusion has been eliminated.

The manual is designed to work hand in hand with the Sega Basic Operators' Manual, and explains many areas which may currently have been somewhat difficult to follow. All in all, by completing the program and following the books, you are sure to become extremely competent in many aspects of working with the computer, whilst enjoying yourself; creating an entertaining game.

# Program Dissection

**by Phil Kenyon**

The following program is a basic game, involving a UFO flying across the screen, and a laser gun at the base of the screen, whose movement is controlled by the operator, using the left and right arrow keys. To shoot down the Space Ship you must fire the laser at the precise moment when it is above you.

By dissecting this program, you will learn the commands to control objects on screen, and monitor for relative positions of other randomly moving objects to gain good understanding of games programming.

Line 10:-

Rem is used purely as a reference for the programmer so that any part of a program can be easily located. All text following a Rem Statement is ignored by the computer.

Line 20:-

Puts the computer into its Graphic Screen mode (the drawing Screen, Mag 1, groups four patterns together to form one sprite, see enclosed section on sprites).

Line 30:-

The program now performs a subroutine on line 130, which when completed, sends back to carry out the next command on line 40.

Line 130:-

To clearly see how the program runs, this dissection should follow this subroutine, which as the Rem suggests, initialises the program by entering in values to certain variable addresses, and designing the shapes or patterns to use as sprites.

```
10 REM ***** UFO GAME *****
20 SCREEN 2,2:MAG 1
30 GOSUB 130
40 IF MN=0 THEN 530
50 K$=INKEY$
60 IF K$=CHR$(28) AND MH+16<215 THEN M
H=MH+8:GOSUB 610
70 IF K$=CHR$(29) AND MH-1>43 THEN MH=
MH-8:GOSUB 610
80 GOSUB 320
90 FX=FX+SP:GOSUB 640
100 IF FX>235 THEN FX=0
110 IF K$=CHR$(30) THEN GOSUB 350
120 GOTO 40
130 REM ::::: INITIALIZE :::::
140 CLS
150 SC=0:MN=30:SX=210:SY=85
160 PATTERN S#0,"0303030303030303"
170 PATTERN S#1,"070F1F3F7FFF3C3C"
180 PATTERN S#2,"8080808080808080"
190 PATTERN S#3,"C0E0F0F8FCFE7878"
200 PATTERN S#4,"0000000000000307"
210 PATTERN S#5,"0F0F7FFF9999FF38"
220 PATTERN S#6,"00000000000C0E0"
230 PATTERN S#7,"F0F0FEFF9999FF1C"
240 LINE(43,70)-(43,189),4
250 LINE(215,70)-(215,189)
260 MH=101:GOSUB 610
270 COLOR6:CURSOR 218,75:PRINT"SCORE"
280 CURSOR 12,75:PRINT"POWER"
290 LINE(25,93)-(27,93+MN*3),,BF
300 RETURN
310 REM ::::: START POSSITION <UFO>:::
::
320 FY=INT(RND(1)*55)
330 SP=INT(RND(1)*15)+5
340 RETURN
350 REM ::::: FIRING :::::
360 MN=MN-1
370 BLINE(25,93+(MN+1)*3)-(27,93+MN*3)
,,BF
380 LINE(MH+7,170)-(MH+7,0),6
390 BEEP:FOR I=1 TO 300:NEXT I
400 BLINE(MH+7,170)-(MH+7,0)
410 IF MH+7<FX OR MH+7>FX+15 THEN RETU
RN
420 BEEP
430 LINE(FX,FY+5)-(FX+15,FY+20)
440 LINE(FX,FY+20)-(FX+15,FY+5)
450 FOR I=1 TO 300:NEXT I
460 BLINE(FX,FY+5)-(FX+15,FY+20)
470 BLINE(FX,FY+20)-(FX+15,FY+5)
480 GOSUB 670
490 SX=SX+10:IF SX=250 THEN SX=220:SY=
SY+10
500 COLOR1:CURSOR SX,SY:PRINT" "
510 SC=SC+10:FX=0
520 RETURN
530 REM ::::: GAME OVER :::::
540 CURSOR 70,80:PRINT"*** GAME OVER *
**"
550 CURSOR 75,100:PRINT"YOUR SCORE : "
;SC
560 CURSOR 75,110:PRINT"HIGH SCORE : "
;HS
570 CURSOR 10,130:PRINT "PRESS THE SPA
CE BAR TO PLAY AGAIN !!"
580 IF INKEY$<>" " THEN 580
590 IF SC>HS THEN HS=SC
600 FX=0:GOTO 30
610 REM ::::: MOVE :::::
620 SPRITE 0,(MH,172),0,11
630 RETURN
640 REM ::::: MOVE <UFO> :::::
650 SPRITE 1,(FX,FY),4,1
660 RETURN
670 REM ::::: <UFO> :::::
680 SPRITE 1,(FX,FY),4,0
690 RETURN
```

Line 140:-

Clears the Graphic Screen ready for us to design our game.

Line 150:-

Remember how with LET we could assign a variable value to a letter, just like putting information into a filing cabinet, and labelling the drawer as "A", or "SCORES", whatever stands for the information kept in the drawer; on this line, we are creating filing cabinets with labels, and storing information in there, therefore "SC" will stand for "SCORE", and so far the score is nil, therefore currently SC$^{5}$0; "MN" represents the number of shots per game allowed, screen coordinates are loaded into "six" and "sy", and their function will become clear later on.

Line 160-230:-

Designs the pattern for the Space Ship and laser gun. (See sprites).

Line 240-250:-

Draws two vertical lines on each side of the screen in blue. This will be the border of the game area.

Line 260:-

Variable MH is created, and 101 is stored there. This will be used as a coordinate to position the laser beam when firing. We must now jump to line 610.

Line 610:-

Line 610 tells us we are about to set up the movement routine.

Line 620:-

Here we assign patterns 0-3 to Sprite 0 (remember MAG 1). This is then positioned at screen location (MH, 172) where MH is currently 101, therefore our laser gun is positioned centrally at the bottom of the screen.

Line 630:-

Returns us to line 260 where we set off from for this routine. To continue on from the next command which in this case, is on line 270.

Line 270:-

We are still in the stage of setting up the screen and this line prints the word "SCORE" in red at screen location 218 dots across, 75 down.

Line 280:-

Positions the word power on the left of the screen in the same colour.

Line 290:-

This draws a diagonal line in the area of the screen below the word power, which has a variable for its last coordinate. The variable is linked to the number of times a shot is fired, causing this amount to increase each time you shoot. As this coordinate increases the size of the line reduces upwards, signifying power loss. The BF turns the diagonal line into a Box or Bar, filled with colour, to give a more readable gauge.

Line 300:-

Returns us to our original GOSUB on Line 30, where we continue on Line 40.

Line 40:-

Checks to see if we have any shots left. If not, then we jump straight to the Game over Routine at line 530. If we do, then continue to line 50.

Line 50:-

Tells the computer to look for a specific key being pressed and whatever key is pressed, to call it K$.

Line 60:-

All the characters and functions of the keyboard and keys have a character code number. These codes are referred to in pages 18, 19, 154, and 155 of the Sega Operators' Manual. The codes for right and left cursor movement, are CHR$ 28 and 29, respectively. Therefore, here the program tells the computer if the right arrow key is being pushed and MH, which is the position of our gun + 16 places to the right, is less than 215, then we must add eight places to our position, which in effect, moves up to the right. The reason it checks the first 16 places is to be sure that we are not trying to move out of the playing area. Remember in line 250, we drew a vertical line which is our border. Its coordinate is 215 across, which we are now not allowing our gun to cross, as explained in issue 1. If the result of IF, THEN is true, we continue reading the line of command, which here would send us to line 610, which carries out the movement of our gun. If the movement is false, we carry on to line 70.

Line 70:-

Checks now to see if the left arrow key is being passed. This checks to make sure we do not try and move outside the left boundary and if it is being pushed, and we can move, 8 is deducted from our value MH, and the program jumps to line 610 to execute the move. IF neither of these keys are being pushed, the program does not wait until one of them is pressed unless you specifically give a command which makes it do so, (i.e. 80 go to 60, which keeps it looking), it will simply do what is on the next line of command.

Line 80:-

If neither key has been pushed, the program now jumps to the subroutine on line 320.

Line 320:-

Here a random number between 1 and 55 is selected by the computer, and stored as FY. INT makes sure the number is a whole number, not a fraction.

Line 330:-

Loads a random number between 5 and 20 into SP. These two variables are used to position the Spaceship.

Line 340:-

Returns us to line 90.

Line 90:-

A new variable address is now created, FX, as we currently have not put anything in FX, it stores 0 nothing, so when we add FX + SP, the result for the moment will have FX and SP being the same. We now jump to line 640 to position our Spaceship.

Line 650:-

Allocates patterns 4-7 to sprite 1 and takes the coordinates from FX, and FY, to position it on screen in blue. The fact that both FX and FY are figures derived from random number generation, means its position will change in an erratic unpredictable manner, making it more difficult to shoot.

Line 660:-

Line 660 returns us to line 100.

Line 100:-

Once our coordinate FX gets too large, there is a danger it might exceed the screen limitation of 255 pixels across, which would result in an error, causing the program to crash. To avoid that happening, and to bring the Spaceship back for another run, the computer is told if FX is greater than 235, then bring it back to 0, which will cause it to start again from the left.

Line 110:-

Now we have positioned ourselves and the Spaceship and set everything in motion, we can try and shoot at it. To fire we are going to use the code for the up arrow key, CHR$(30). If this is being pushed at this part of the program, jump to 350, if not, go back to 40 and start the laser and spaceship movement routine again.

Line 350:-

Here is the Firing Routine.

Line 360:-

First deduct one from the number of remaining shots.

Line 370:-

Erases a small amount of the power bar by using the decreasing value of MN as a coordinate for the length of the line.

Line 380:-

Draws a line from MH, which is the position coordinate of our laser, +7 to make the line come from the centre of the sprite, not the side. (Remember in Mag 1, we have a 16 x 16 matrix sprite counting 0-15 makes 7 or 8 the centre dot), and 170 down, which is the bottom line coordinated of our laser; to the same point at the top of the screen, i.e. (MH + 7) – (MH + 7,0).

This line is drawn in red and represents a laser beam.

Line 390:-

Makes a beep and the line is held for a delay of 1 to 300 (about two seconds).

Line 400:-

The line is now erased from the screen.

Line 410:-

Checks to see whether the line had made contact with the Spaceship by comparing the relative positions of the centre of your laser, and the position of the spaceship. The two coordinates which must match, are the ones which position objects across the screen, which are MH for the laser, and FX for the spaceship, so by taking MH + 7 Laser beam location, and saying if it is less than FH, you have missed, or if MH + 7 is greater than FX+15, i.e. the full span of dots covered by the spaceship, you have also missed. If not, then you must have hit. Therefore, if you missed, return to line 120 and try again. If it is a hit, you continue with Line 420.

Line 420:-

Another Beep.

Line 430-440:-

Draws two diagonal lines using the coordinates of the spaceship to draw a cross through the ship indicating a hit.

Line 450:-

Count from 1 to 300.

Line 460-470:-

Erase the cross.

Line 480:-

Jump to routine one line 670 which puts the same spaceship sprite over the existing sprite in a transparent colour, causing it to disappear, before returning to line 490.

Line 490:-

SX and SY position the small graphic of the Space Invader on the right score area, the coordinates are increased to position them in rows.

Line 500:-

Prints the Space Invader character on screen coordinates SX, SY, in black.

Line 510:-

Adds 10 to the score and sends the spaceship back to the beginning coordinate.

Line 520:-

Returns us to line 120, which sends us to line 40.

The remainder of the program should now be self-explanatory, and should give you a good understanding of how to begin creating your own more sophisticated programs.

Begin by altering some of the variables governing control in this program, to see how it affects movement. For instance, increase the range of random numbers you can choose from, or the size of steps your laser gun can take.

You should also try and put sound commands in place for Beep for the laser gun and explosions. You could finish up with something completely out of this world!

Finished Facia of Sega Surer Control Station released soon

# COMPUTER EDUCATION A Nation's Responsibility

**by Mr Fenton MANAGING DIRECTOR Grandstand Leisure**

The New Zealand community at large stands at an unprecedented crossroads.

At a time when more and more people are claiming that the education system is failing an increasing number of children, there is a means to reverse the process.

It is the computer: The first tool ever which seems capable of motivating any child to learn, even those who have shown no previous interest or aptitude.

It works for a combination of reasons:

It is fun; enjoyable learning is always more effective than learning by drudgery.

It will go over and over the same ground repeatedly until even the slowest learner has "got the hang of" the subject. Neither the best of teachers nor the best of parents has that capacity and often it is that time input and patience which will turn an under-achiever into an achiever.

It offers the gratification of instant results to boost the morale of even the most disillusioned child, encouraging them to work on and one and in the process, build self-esteem.

The computer works in a totally logical manner and that seems to appeal to children, particularly young ones.

It is challenging in a manner that children seem to universally respond to.

In operating a computer children are required to use other skills, such as reading, comprehension and arithmetic, so it also hones these attributes "in passing."

The computer can also be a massive information library. With the right programme and a big enough memory, it can offer the answer to virtually any question at the touch of a button.

Unfortunately, it will be a long time before the school system can offer adequate and equal computer time for these benefits to be enjoyed by all children.

Children fortunate enough to have a home computer will, therefore streak even further ahead of disadvantaged children, than would have been the case before computers were used in education: possibly further than at any time since the introduction of universal education.

That is the paradox of the computer. It is the first ever means which can motivate any child to become an achiever, but if it is not made widely available, the "haves" will have even more, and the "have-nots" will have even less.

In short, those children who are already educationally disadvantaged will be even more disadvantaged as a result.

In November 1982, Time Magazine carried a report on the availablitiy of computers in American schools. It was titled "Peering Into The Poverty Gap" and subtitled "Will the rich get smarter while the poor play video games?"

It stated that in the US rich schools were getting computers and poor ones were not, and quoted the president of Market Data, Herbert Lobsenz, as saying: "If computers are the wave of the future, a lot of America is being washed out."

It added later that some observers saw the computer as a potential educational levelling device. Computer consultant Charles Lecht was quoted: "Students who used to fail because they could not master geometry the first time around, will be able to turn to the computer for relief. The machines will emerge as the great equalizers."

While this report does not make a direct comparison with home computers in New Zealand, the principle is the same: Those who get computer time, get ahead.

Compounding that educational dilemma, is the careers problem of the immediate future.

With the advent of the micro-computer it is possible for every business to be computerised. This, together with sales of home computers, has created for the first time a mass market for computers. With that has come the economies of scale and the competitive pressures to enable computers to develop faster than ever before.

It can be safely predicted that within ten years, there will be few if any, jobs which are not either computer based or at least computer assisted. That, in turn, means that the best "computer educated" children of today, will have the pick of tomorrow's jobs. "Computer illiterates" will be left with the scraps and those scraps will include fewer manual jobs, the traditional source of employment for the non-scholar.

To quote again from Time, reporting Andrew Molnar, computer education specialist at the (US) National Science Foundation: "Power is not distributed evenly now, and computers will broaden that gap."

What this means is that there is now a new dimension to education. Not only must all children use computers to help them learn, and to master traditional subjects; but they must also study computers, computer software, computer programming and so on, as a separate subject — like English, or typing or metal-work.

Children must know about computers from the earliest age, because tomorrow's job market will make "computer competence" as essential as English.

Time quoted another computer company executive, Arden Bement saying: "It's a new way of thinking. The kids who don't get indoctrinated to computers by seventh grade are not going to develop the same proficiency."

What has happened is that we have finally entered the "computer age." It has been a long time coming, but now that it is here there is the risk that unless it is managed properly, it could engulf the entire present generation of infants, spit out the traditionally disadvantaged and leave them worse off than they have ever been, both in terms of general education and of job potential; and it will reduce a bigger proportion of children than ever, to the ranks of the disadvantaged.

Properly managed, on the other hand, it has the capacity to offer children opportunities that only a few years ago would have been thought impossible.

Children that have a computer will be at an advantage over those who only get computer time at school; and at a massive advantage over those who do not get any computer time at all.

The ideal answer would be to have a micro in every home, but that is an unrealistic hope for quite a few years yet. As a result certain children, now born, are going to be put at a bigger and bigger

disadvantage that may cripple them for the rest of their lives.

There are four reasons why not all children will get a home computer and only two of them are easily solved by educating the parents.

There are parents who care about their children but who regard home computers as nothing but "rich kids' toys." There are also parents who believe that the present education system is letting down their children for cultural reasons.

The first group can probably be persuaded to consider a computer once they have the true facts at their disposal.

The second group can be likewise persuaded once it is demonstrated that the computer is the original multi-cultural teacher; a device which can be programmed to educate in their way, and provide the support that seems to be lacking now. Then, however, there is poverty, or relative poverty.

It will soon be possible for children of poor parents who may want to have a computer but cannot afford it, to slide into the educationally deprived category for purely economic reasons.

This has not been the case before in this country but if it does happen it will be because of a gap between the advance in technology and the state's ability to provide an adequate response in the short term.

Then there are parents who do not care about their children's education, who believe the system is against "people like them," or who have insufficient education themselves to appreciate its value.

Their children are traditionally the biggest losers because they have always lacked the encouragement and motivation to even try.

They stand now to be even bigger losers. For them, the computer represents the first ever tool which actually acts to break that vicious cycle and give them for the first time, the potential and encouragement to achieve, despite the circumstances at home.

Perversely, their parents are highly unlikely to provide them with the tool, so they face the prospect of increased deprivation.

As the state cannot solve this dilemma in the short term, and the parents cannot or will not, the only answer is for concerned groups whether they be service clubs, maraes, or just groups of like-minded neightbours, to pool their resources to form a local computer club or at the very least, to provide a computer or two for the neighbourhood children.

While that solution might sound naive and idealistic, there are plenty of precedents. Such orgainsations as Boystown, the YMCA, Scouts, Red Cross, St Johns, and dozens of other sports, recreational, and activity clubs, were all set up by concerned citizens to offer children healthy, constructive facilities that they would not have otherwise been able to enjoy.

There is no difference in principle between them and the suggested computer or computer and video games clubs.

While all this must read like a comercial from home computers, it is no more so than saying children need books and pencils for their schooling; or that a fire engine is needed if a house is burning down.

The problem is probably only a relatively short term one. A few years from now, perhaps no more than ten years, the school system may well have placed a terminal on every desk; but until that happens the "have-not" are at an immense disadvantage. As the Time report stated: " . . . the majority . . . worry about the near-term spectre of the rich taking control of the technology while the poor play video games."

The question is, are we going to tackle the problem as a community or not? Are we going to act to fill the gap ourselves in the meantime? Or are we just going to sit back and let perhaps the best opportunity in the history of education to bridge all the system's shortcomings, slip by?

Grandstand are for our part doing our best to make as many people as possible aware of the importance of being computer literate. We will also offer our services to any Fund Raising association who are prepared to assist us in supplying computers to any institution in your area that would benefit from using Sega computers.

To establish the ways in which we could work together toward this goal, please write with details of your association, and the areas of the community which you feel would be able to help, to

**GRANDSTAND**
**Box 2353**
**AUCKLAND**


## SEGA PROGRAMMERS MANUAL

**By B. BROWN.**

This manual is written with the more serious programmer in mind, and contains a wealth of information concerning sound graphics, and machine code information, sure to be of interest to the Sega enthuasiast.

Normal retail price will be $29.95. To obtain your copy for $24.95 cut out enclosed coupon and send to.

NAME: .....

ADDRESS: .....

.....

..... PH .....

**NO STAMP REQUIRED**

I enclose payment: ✓

☐ Visa ☐ Bankcard

☐☐☐☐☐☐ ☐☐☐☐☐☐ ☐☐☐☐☐☐

Cardholder signature.....

Date card expires ..... ..... .....

Cheque ☐ Cheque No.....

Postal Order ☐ Postal note ☐

**GRANDSTAND**
**Box 2353**
**AUCKLAND**

# MUSIC CARTRIDGE REVIEW

**Philip Bachler**

The Sega Music Cartridge has been available now for some time for the Sega computer. EFFECTIVELY the cartridge converts the Sega keyboard into the keyboard of a musical instrument, with an overlay card which assigns notes to the keys and the screen then displays the notes as they are passed by the operator as sheet music.

This cartridge is extremely easy to operate, and is not only great entertainment, but extremely educational as it teaches you how to play music, read and write music, even arrange music. So far, as it is obviously easier to demonstrate Games, the cartridge has suffered from a lack of exposure, which we now hope to overcome by sharing the wide range of options available, as explained in the extensive operating instructions contained in the manual which is supplied with every cartridge.

When you first turn the computer on, you are shown the music editor cartridge title screen, a short tune is played. The computer then displays the music menu with 9 options. This makes using this cartridge very easy as all you have to do is follow the screen prompts. The easiest way to explain merits of this cartridge is to go through this menu.

## OPTION 1

Displays Music Titles. This prints what pieces of music you have stored for the memory. This cartridge is very clever in the way that it can store approximately 20 minutes of music in its memory. Depending on the length of each piece of music you store, more than 30 songs at once (Of course you can have as many pieces as you like on cassette).

## OPTION 2

Composition: This option allows you to enter and store musical information in the computers memory, which can be played, edited on cassette, when you select certain pieces of information before you can key in the music. Firstly the computer asks you for the name of the piece of music that you wish to enter into the computers memory from the keyboard. This name maybe up to 32 characters long and include upper and lower case, full punctuation and numbers. This allows you to fully label and name every piece of music you create, preventing confusion when playing, saving and editing the music.

You are prompted for the key signature of the piece of music (how many sharps and how many flats the piece of music has). This is a real time saving feature as when you are keying in the music you are reminded that certain notes are sharp and flat. It also allows you to play the piece of music in any key.

The computer requires the tempo of the piece of music. This is very useful as you can slow a piece down and play along with another instrument. It is also quite amusing to hear a piece of music being played at 2 or 3 times the speed it should be.

Once the computer has received all the information it requires you can input musical data. It moves to a graphic screen with two musical staves and a piano keyboard. Now the user can compose a piece of music or copy a piece of sheet music into the computers memory. All you have to do is press the note or notes you want to play and tell the computer how long you want them played for. This makes entering and creating music very easy, as you do not require any mathematical formula or complicated musical knowledge.

The music cartridge is very good at editing. Using the arrow keys you can move through the music in the memory, inserting, deleting and changing notes.

The music editor cartridge has a test function. This plays the music currently on the screen and is very helpful as you do not have to play a whole piece of music to see if one or two notes sound alright. The test function really helps if your composing your own music, as it allows you to hear small sections of your piece of music while you are creating it. When you have finished keying in the musical data you can return to the main menue simply by pressing the Break key. The music in the memory is not damaged in any way.

## OPTION 3 ARRANGEMENT

You can alter any information for any piece of music with this option. The user can change the name of the piece of music, its speed or any note in the music. It is also very easy to transpose (move) the whole piece of music up and down the musical stave. This means if you do not like how the music is being played you can alter anything until the tune is just right.

## OPTION 4 PLAYING THE MUSIC

This option will play any number of pieces of music in the computers memory as many times as you like. The advantage of this is that you can set and forget. The computer will play the music all day if you want it too.

## OPTION 5

With this option you can choose between three different instruments (piano and harpsichord and a organ) the tune may sound totally different when it is played on an organ instead of on a piano, so it is possible to get a different and unique sound with every tune.

## OPTION 6 SAVING MUSICAL INFORMATION ON CASSETTE

Any tune in the computers memory can be saved on cassette, so that you can keep it forever. You can load it back into the computers memory from a cassette to play or alter it whenever you want too. After a short while you will build up a small library of tunes and compositions that you have saved on cassette.

## OPTION 7 LOADING MUSICAL INFORMATION FROM CASSETTE

It is impossible to load pieces of music from the cassette directly into the computers memory. This means that every time you want to play a tune, you do not have to type it all in again.

## OPTION 8 VERIFYING

This option allows you to see a piece of music that has been recorded correctly on the cassette. If it does not verify you can simply re-record the tune on cassette. This helps prevent tape reading errors when loading music into the computers memory.

## OPTION 9 DELETING

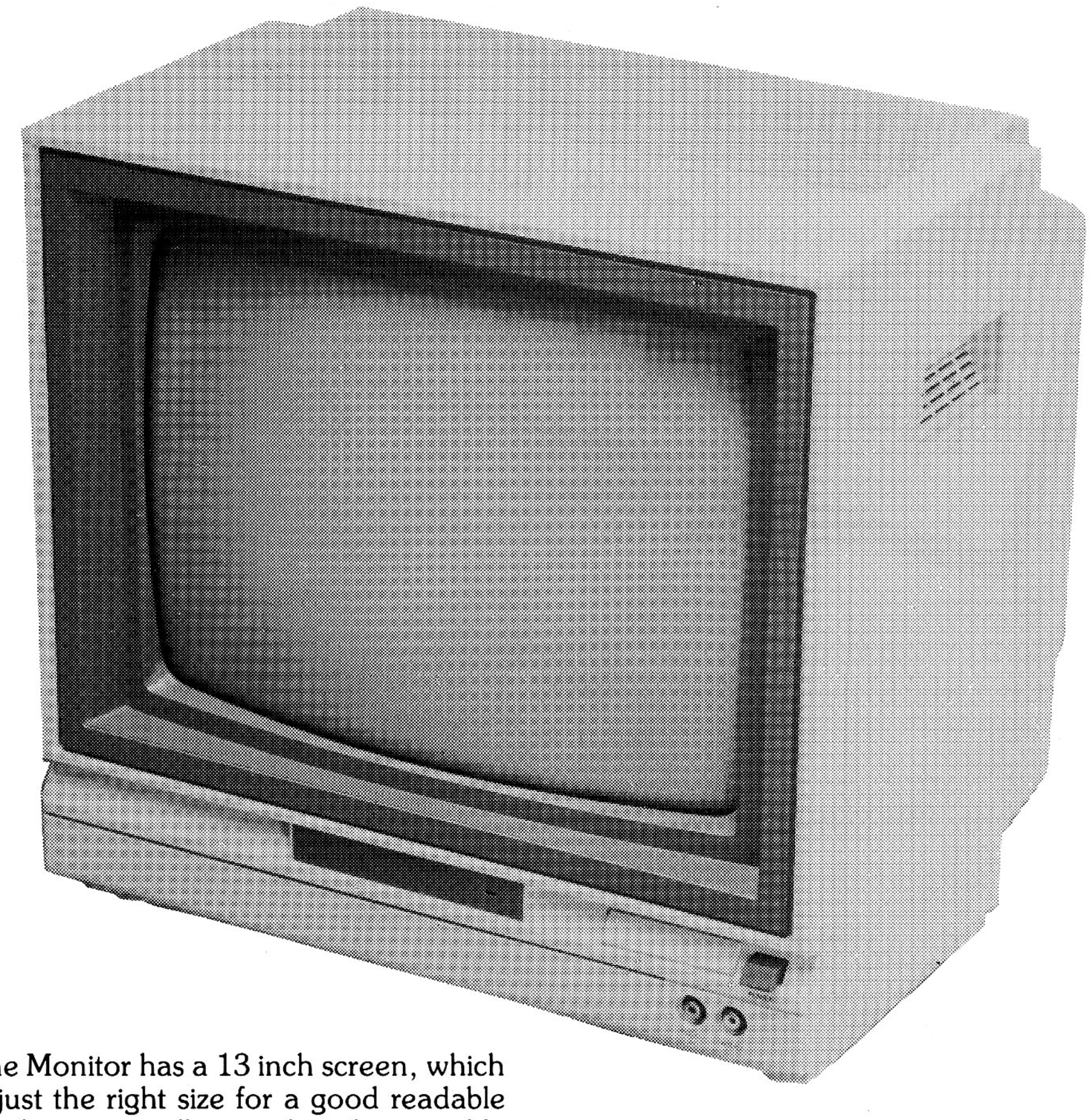
The computer prints the titles of all the pieces of music in its memory and asks the user which tunes are to be deleted from the memory. This gives you more room to put in new pieces of music, at the same time as getting rid of old and unwanted pieces of music. As you can see the music editor cartridge is very easy to use and you do not need any real musical knowledge to operate it properly. There is a 30 page manual included with each cartridge that explains all the music cartridge functions and depths.

Grandstand has produced a music cartridge demonstration cassette that contains 10 pieces of music that can be loaded into the computer. These tunes show just what the music cartridge is capable of producing. This cassette will play:

Hooked on computer classics Parts 1-2
The Entertainer
Hark! The Herald angels sing
Beethoven Minuet in G
The Sailors Horn Pipe
Yume-No-Tochyu
Nocturne
Rydeen
Beautiful Dreamer
From Russia with Love

The cassette insert contains information and hints on how to get from starting with the music editor cartridge to producing music of high quality. There is also a quick reference guide listing the functions of many of the keys. This cartridge is a very polished piece of educational software that will provide hours of fun and enjoyment, while at the same time teaching you and your family basic musical skills.

# SEGA
# COLOUR MONITOR
# REVIEW



**Are you satisfied with the quality of your display? Are you considering buying a second T.V. to use with your computer?**

**. . . DON'T**

As you can now get a high quality, low cost SEGA Colour Monitor. This monitor has been designed to operate specifically with the SEGA, so you are assured of a crisp, clear picture.

A Colour Monitor has many significant advantages over a television:

1. You get a sharp, clear undistorted picture. This is due to the fact there is no tuning unit in the monitor. The information from the computer is passed straight through to the screen.
2. The Colour Monitor has a speaker built into it. The advantage of this is you do not have any hiss and white noise interference.
3. It also can be used with Video Recorder or player.

The Monitor has a 13 inch screen, which is just the right size for a good readable display yet small enough to be portable (there are even handles built into the side of the case to make it easy to carry).

The user has full control over the picture quality. There are tint, colour, brightness, contrast, vertical hold, horizontal position and volume controls.

One of the greatest advantages a Colour Monitor has is that it allows you to place the computer in a more convenient place, rather than in your sitting room with your T.V. It also means that some people can be watching T.V. while others can be programming with the computer elsewhere.

There is only a small shipment of these high quality SEGA Colour Monitors arriving in New Zealand in December. A number of these have already been ordered, so if you are interested please complete the form below and return to us poste haste.

Yours sincerely

Philip Kenyon
Computer Division

**Dear Phil**

**I am interested in purchasing a SEGA Colour Monitor for $750.00. Please contact me when they arrive from SEGA and I will send you my cheque.**

**Name** . . . . . .

**Address** . . . . . .

. . . . . .

Signature . . . . . .

To purchase the music editor or demonstration cassette visit your local dealer or send your cheque to:

**Sega Users Club**
**P.O. Box 2353**
**Auckland**

Music editor cartridge $125.00
Demonstration cassette $ 9.95

VISA

MasterCard

or phone with Visa/Bankcard
09-504-033

# This Program Designs the Manhattan Skyline

TRY PUTTING YOUR OWN GAME ON THIS!

```
10 SCREEN 2,2:CLS
20 X=0:Y=10:XX=255:YY=191:E=1
30 COLOR 15,5,(0,0)-(255,49),1
40 FOR N=70TO 0 STEP-10
50 LINE(X,Y-E)-(XX,Y-E),15
60 E=INT(SIN(RAD(N))*10)
70 Y=Y+E
80 NEXT N
90 COLOR1,1,(0,50)-(255,99),1
100 COLOR1,1,(0,101)-(255,191),1
110 X=0:Y=100:XX=255:YY=191:E=1
120 COLOR  15,1
130 FOR N= 0TO 70STEP  10
140 E=INT(SIN(RAD(N))*20)
150 LINE(X,Y+E)-(XX,Y+E),15
160 Y=Y+E
170 NEXT N
180 REMLINE(56,100)-(64,70),14,BF
190 REMLINE(68,100)-(80,60),1,B
200 REMLINE(74,100)-(84,80),9,BF
210 GOSUB 250
220 END
230 GOTO  430
240 SCREEN 2,2:CLS
250 X=80:Y=100:XX=0:YY=191
260 FOR U=0 TO 8
270 LINE(X,Y)-(XX,YY),15
280 X=X+10:XX=XX+30
290 NEXT U
300 X=70:Y=100:XX=0:YY=160:E=5
310 FOR U=0 TO 2
320 LINE(X,Y)-(XX,YY),15
330 X=X-10:YY=YY-20
340 NEXT U
350 LINE(40,100)-(0,110),15
360 X=170:Y=100:XX=255:YY=175:E=5
370 FOR U=0 TO 2
380 LINE(X,Y)-(XX,YY),15
390 X=X+10:YY=YY-20
400 NEXT U
410 LINE(200,100)-(255,120),15
420 LINE(220,100)-(255,110),15
430 LINE(50,80)-(53,71),15:LINE-(57,71
):LINE-(58,75):LINE-(58,63):LINE-(62,6
3):LINE-(62,75):LINE-(65,75)
440 LINE(65,75)-(68,73):LINE-(76,73):L
INE-(76,71):LINE-(79,71):LINE-(79,61):
LINE-(82,55):LINE-(86,66):LINE-(86,77)

450 LINE(86,77)-(87,75):LINE-(89,79):L
INE-(89,81):LINE-(92,81):LINE-(92,77):
LINE-(93,77):LINE-(93,74):LINE-(95,74)

460 LINE(95,74)-(95,59):LINE-(100,59):
LINE-(103,61):LINE-(103,68):LINE-(107,
68):LINE-(108,69):LINE-(108,73)
470 LINE(108,73)-(113,73):LINE-(113,68
):LINE-(118,68):LINE-(118,52):LINE-(12
5,52):LINE-(125,78):LINE-(129,76):LINE
-(129,76)
480 LINE(129,76)-(129,53):LINE-(135,53
):LINE-(135,81)
490 LINE(0,61)-(0,61):LINE-(3,69):LINE
-(8,69):LINE-(8,63):LINE-(13,62):LINE-
(27,65):LINE-(27,80):LINE-(33,80)
500 LINE(33,80)-(33,78):LINE-(43,79):L
INE-(49,78):LINE-(49,80):LINE-(50,80)
510 LINE(135,80)-(140,75):LINE-(145,65
):LINE-(150,65):LINE-(150,75):LINE-(15
5,75):LINE-(155,70):LINE-(160,70)
520 LINE(160,70)-(160,60):LINE-(165,60
):LINE-(165,65):LINE-(170,65):LINE-(17
0,70):LINE-(179,69):LINE-(185,70)
530 LINE(185,70)-(185,72):LINE-(186,72
):LINE-(189,63):LINE-(193,63):LINE-(19
3,67):LINE-(195,67):LINE-(195,66)
540 LINE(195,66)-(198,66):LINE-(198,67
):LINE-(204,67):LINE-(204,65):LINE-(21
2,65):LINE-(212,63):LINE-(216,63)
550 LINE(216,63)-(216,68):LINE-(222,68
):LINE-(222,69):LINE-(226,69):LINE-(22
6,72):LINE-(228,72):LINE-(228,52)
560 LINE(228,52)-(233,52):LINE-(237,54
):LINE-(237,61):LINE-(241,61):LINE-(24
1,67):LINE-(246,67):LINE-(246,61):LINE
-(255,61):PAINT(0,55),8
570 FOR I=0 TO 300
580 X=INT(RND(1)*255)
590 Y=INT(RND(1)*100)
600 IF Y<61 THEN 590
610 PSET(X,Y),8:NEXT I
```

# GLOSSARY

**Accessory Devices** - additional equipment which attaches to the computer and extends its functions and capabilities. Included are preprogrammed cartridges* and units which send, receive or store computer data, such as printers and disks. These are often called peripherals.

**Array** - A collection of numeric or string variables, arranged in a list or matrix for processing by the computer. Each element in an array is referenced by a subscript* describing its position in the list.

**ASCII** - The American Standard Code for Information Interchange, the code structure used internally in most personal computers to represent letters, numbers, and special characters.

**BASIC** - an easy-to-use popular programming language used in most personal computers. The word BASIC is an acronym for "Beginners All purpose Symbolic Instruction Code."

**Baud** - commonly used to refer to bits per second.

**Binary** - a number system based on two digits, 0 and 1. The internal language and operations of the computer are based on the binary system.

**Branch** - a departure from the sequential performance of program statements. An unconditional branch causes the computer to jump to a specified program line every time the branching statement is encountered. A conditional branch transfers program control based on the result of some arithmetic or logical operation.

**Breakpoint** - a point in the program specified by the STOP command where program execution can be suspended. During a breakpoint, you can perform operations to help you to locate program errors. Program execution can be resumed with a CONT command, unless editing took place while the program was stopped.

**Bug** - a hardware defect or programming error which causes the intended operation to be performed incorrectly.

**Byte** - a string binary* digits (bits) treated as a unit, often representing one data character*. The computer's memory capacity is often expressed as the number of bytes available. For example, a computer with 16K bytes of memory has about 16,000 bytes available for storing programs and data.

**Cartridges** - preprogrammed ROM* modules which are easily inserted in the SEGA computer to extend its capabilities.

**Character** - a letter, number, punctuation symbol, or special graphics symbol.

**Command** - an instruction which the computer performs immediately. Commands are entered with no preceding line number.

**Concatenation** - linking two or more strings* to make a longer string. The "+" is the concatenation operator.

**Constant** - a specific numeric or string* value. A numeric constant is any real number such as 1.2 or -9054. A string constant is any combination of up to 248 characters enclosed in quotes, such as "HELLO THERE" or "275 FIRST ST."

**Cursor** - a symbol which indicates where the next character* will appear on the screen when you press a key.

**Data** - basic elements of information which are processed or produced by the computer.

**Default** - a standard character or value which the computer assumes if certain specifications are omitted within a statement* or a program*.

**Device** - (see Accessory Devices)

**Disk** - a mass storage device capable of random and sequential access.

**Display** - (noun) the video screen; (verb) to cause characters to appear on the screen.

**Execute** - to run a program; to perform the task specified by a statement* or command*

**Exponent** - a number indicating the power to which a number or expression* is to be raised; usually written at the right and above the number. For example, $2^8 = 2x2x2x2x2x2x2x2$. In SEGA BASIC the exponent is entered following the letter "E" in scientific notation*. For example, $2 = 2\ 8; 1.3 \times 10 = 1.3E25$.

**Expression** - a combination of constants, variables, and operators which can be evaluated to a single result. Included are numeric, string, and relational expressions.

**File** - a collection of related data records stored on a device; also used interchangeably with device* for input/output equipment which cannot use multiple files, such as a line printer.

**Function** - a feature which allows you to specify as "single" operations a variety of procedures, each of which actually contains a number of steps; for example, a procedure to produce the square root via a simple reference name.

**Graphics** - visual constructions on the screen, such as graphs, patterns, and drawings, both stationary and animated. SEGA BASIC has build-in subprograms which provide easy to-use colour graphic capabilities.

**Hardware** - the various devices which comprise a computer system, including memory, the keyboard, the screen, disk drives, line printers, etc.

**Hertz(HZ)** - a unit of frequency. One Hertz = one cycle per second.

**Hexadecimal** - a base .16 number system using 16 symbols, 0.9 and A-F. It is used as a convenient "shorthand" way to express binary* code. For example, 1010 in binary = A in hexadecimal, 11111111 = FF. Hexadecimal is used in constructing patterns for graphics characters in the PATTERN subprogram.

**Increment** - a positive or negative value which consistently modifies a variable*.

**Input** - (noun) data* to be placed in computer memory; (verb) the process of transferring data into memory.

**Input Line** - the amount of data* which can be entered at one time. In SEGA BASIC, this is 248 characters.

**Internal date-format** - data* in the form used directly by the computer. Internal numeric data is 8 bytes* long plus 1 byte which specifies the length. The length for internal string data is one byte per character in the string* plus one length-byte.

**Integer** - a whole number, either positive, negative or zero.

**I/O** - Input/Output; usually refers to a device function. I/O is used for communication between the computer and other devices (e.g. keyboard, disk)

**Iteration** - the technique of repeating a group of program statements; one repetition of such a group. See Loop.

**Line** - see input line, print line, or program line.

**Loop** - a group of consecutive program lines which are repeatedly performed, usually a specified number of times.

**Mantissa** - the base number portion of a number expressed in scientific notation*. In 3.264E + 4, the mantissa is 3.264.

**Mass Storage Device** - an accessory device*, such as a cassette recorder or disk drive, which stores programs and/or data* for later use by the computer. This information is usually recorded in a format readable by the computer, not people.

**Memory** - see RAM, and ROM, and mass storage device.

**Noise** - various sounds which can be used to produce interesting sound effects. A noise, rather than a tone, is generated by the SOUND subprogram* when commands 4 and 5 are specified.

**Null String** - a string* which contains no characters and has zero length.

**Number Mode** - the mode assumed by the computer when it is automatically generating program line* numbers for entering or changing statements.

**Operator** - a symbol used in calculations (numeric operators) or in relationship comparisons (relational operators). The numeric operators are +,-,*,/,. The relational operators are , ,=, =, .,= , =,=

**Overflow** - the condition which occurs when a rounded value greater than 9.9999999999999E+99 or less than -9.9999999999999E-99 is entered or computed. When this happens, a warning is displayed, and the program* stops.

**Output** - (noun) information supplied by the computer; (verb) the process of transferring information from the computer's memory onto a device, such as a screen, line printer, or mass storage device*.

**Parameter** - any of a set of values that determine or affect the output of a statement* or function*.

**Print Line** - a 38-position line used by the PRINT statement.

**Program** - a set of statements which tell the computer how to perform a complete task.

**Program Line** - a line containing a single statement*. The maximum length of a program line is 248 characters*.

**Pseudo-random number** - a number produced by a definite set of calculations (algorithm) but which is sufficiently random to be considered as such for some particular purpose. A true random number is obtained entirely by chance.

**RAM** - random access memory; the main memory where program statements and data* are temporarily stored during program execution*. New programs and data can be read in, accessed, and changed in RAM. Data stored in RAM is erased whenever the power is turned off of BASIC is exited.

**Reserved Word** - in programming languages, a special work with a predefined meaning. A reserved word lmust be spelled correctly, appear in the proper order in a statement* or command*, and cannot be used as a variable* name.

**ROM** - read-only memory; certain instructions for the computer are permanently stored in ROM and can be accessed but cannot be changed. Turning the power off does not erase ROM.

**Run Mode** - when the computer is executing* a program, it is in Run Mode. Run Mode is terminated when program execution ends normally or abnormally. You can cause the computer to leave Run Mode by pressing BREAK during program execution (see Breakpoint*).

**Scientific Notation** - a method of expressing very large or very small numbers by using a base number (mantissa*) times ten raised to some power (exponent*). To represent scientific notation in SEGA BASIC enter the sign, then the mantissa, the letter E, and the power of ten (preceded by a minus sign if negative). For example, 3,264; -2.47E -17.

**Scroll** - to move the text on the screen so that additional information can be displayed.

**Software** - various programs which are executed by the computer, including programs built into the computer. Cartridges* programs, and programs entered by the user.

**Statement** - an instruction preceded by a line number in a program. In SEGA BASIC, more than one statement is allowed in a program line*.

**String** - a series of letters, numbers and symbols treated as a unit.

**Subprogram** - a predefined general-purpose procedure accessible to the user through the statement in SEGA BASIC. Subprograms extend the capability of BASIC and cannot be easily programmed in BASIC.

**Subroutine** - a program segment which can be used more than once during the execution* of a program, such as a complex set of calculations of a print routine. In SEGA BASIC a subroutine is entered by a GOSUB statement and ends with a RETURN statement.

**Subscript** - a numeric expression which specifies a particular item in an array*. In SEGA BASIC the subscript is written in parentheses immediately following the array name.

**Underflow** - the condition which occurs when the computer generates a numeric value greater than -1E-100, less than 1E-100, and not zero. When an underflow occurs, the value is replaced by zero.

**Variable** - a name is given to a value which may vary during program execution. You can think of a variable as a memory location where values can be replaced by new values during program execution.

***See definition in Glossary**


# SOFTWARE FOR SEGA COMPUTER

**CARTRIDGES**

BASIC LEVEL II
BASIC LEVEL IIIA
BASIC LEVEL IIIB
MUSIC
N/SUB
SAFARI HUNTING
BORDER LINE
CONGO BONGO
YAMATO
STAR JACKER
CHAMPION TENNIS
MONACO GP
CHAMPION BASEBALL
SINBAD MYSTERY
VIDEO FLIPPER
PACAR
POP FLAMER
CHAMPION GOLF
EXERION

**HOME/PERSONAL/FINANCE CASSETTES**

FILE SYSTEM
CHEQUE BOOK RECONCILIATION
LOAN & MORTGAGE CALC.
GRAPH & CHART
ACCOUNTS RECEIVABLE
ACCOUNTS PAYABLE
MAILING LIST

**GAME CASSETTES**

TOWERS OF HANIO
HANGMAN
CITY-LANDER
CUBE-IT
BUGALOO
LASER BLAST
MUNCHMAN

**EDUCATIONAL CASSETTES**

ROCKET MATHS
LEARNING ALPHABET
WATCH ME DRAW
SPELLING
ADDITION
SUBTRACTION
MULTIPLICATION
SPRITE EDITOR
MUSIC EDITOR
TYPING TUTOR
LEARNING TO COUNT
SHAPE & COL. QUIZ
MUSIC CARTRIDGE DEMONSTRATION

**Should you find any of these titles difficult to obtain through your stockist, please contact the USER CLUB for mail order details, 504-033.**

# THE UNEXPLAINED SEGA COMMANDS

Ref USERS:COL 4

## Part Two

© B. Brown 1984

The commands dealt with in this section of the magazine are,

SOUND
ERASE
AND
OR
XOR
NOT
CALL

The SOUND Command: The SOUND command is used to generate tones or sounds. The tones are generated by the sound generator chip, which has three independant sound channels and a single noise generator. The format of the sound command is,

SOUND N, F, V

where N is the sound channel number, F is the frequency of the note to be generated, and V is the volume (i.e. how loud) of the note.

The sound channel number: The sega manual lists six different values for the channel number. Taking each in turn,

0 This will turn off all sound channels. The F and V are not to be included, so the format is SOUND 0
1 This refers to sound channel 1
2 This refers to sound channel 2
3 This refers to sound channel 3
4 This is the noise generator set to "white noise"
5 This is the noise generator set to "periodic noise" where the frequency of the noise is determined by sound channel three's frequency.

**The Frequency:** This value determines the frequency of the desired note that you wish to play. The table on page 136 of the basic manual lists the values for each musical note. Let's say that we want to play a note on channel 1 of C#, then the value of F derived from the table is 139, (for octave 2, the higher the octave the higher the note is), so the command is SOUND 1, 139, V where V is a variable between zero and fifteen, which represents how loud the note is to played. Note that you can use all the sound channels at once, they are all independant of each other. In this way, it is possible to play two or more tunes at once.

You can also generate a tone with a specific frequency by specifying F directly, i.e. to generate a tone of 1000hz using tone generator three, the command would be SOUND 3, 1000, V, where V is the volume of the note.

**The Volume:** V is a value between 0 and 15 and represents the loudness of the note. The higher V is, the louder the note is played, so if we want maximum volume then make V equal to 15, i.e. SOUND 2, . . . , 15.

**ERASE:** This command is used to erase the contents of a particular array dimensioned earlier in a program. Let's look at what an array is first, then look at erasing it. An array is very handy for storing data, such as names, addresses, telephone numbers etc. Let's imagine that we have a list of ten names, ten addresses and ten telephone numbers. We need to store this information within a program, so we start off by assigning name one as A1$, we then assign name two as A2$, etc. Now we assign B1$ to the address one, and so on. Notice that there are a lot of strings to keep track of, and it gets rather messy. A better way is to use an array.

First we dimension an array for the names, addresses and telephone numbers. This could be as follows,

```
DIM NM$(10),AD$(10),TN(10)
```

As there are ten items in each of the names, addresses, and numbers then it follows that

```
NM$(1) = first name
NM$(10)= last name
AD$(1) = first address
AD$(10)= last address
TN(1)  = first telephone number
TN(10) = last telephone number
```

It thus becomes very simple to list the name, address and number of any one of the ten people concerned, e.g.

```
FOR X=1 TO 10
PRINT "NAME:    ";NM$(X)
PRINT "ADDRESS:";AD$(X)
PRINT "NUMBER:  ";TN(X)
NEXT X
```

Or to input information into each of the arrays for each person, e.g.

```
FOR X=1 TO 10
INPUT "NAME:    ";NM$(X)
INPUT "ADDRESS:";AD$(X)
INPUT "NUMBER:  ";TN(X)
NEXT X
```

Arrays thus help to organise large sets of information into a menaingful and easy to use format. Erase wipes the information and the array from memory, so that it just doesn't exist anymore, thus it releases the memory space that has been used by that array so that the memory can be used for other purposes. This command will be of great value with the disk drive, but its value is rather limited with just the cassette tape.

**THE LOGICAL OPERATORS:** The commands include the following, AND, OR XOR, and NOT. These commands work in 'binary', that is, they converty the numbers to binary, then do the calculation, then they convert the result back to decimal.

AND: The rules for binary AND operations are that if both bits are a '1' there is a result of a '1', or else the resultant bit is '0'. Let's take an example of two digits and trace the working through,

12 AND 7

Where 12 is 1100 in binary and 7 is 0111, so lets put 12 on the top and seven underneath,

```
12 =   1 1 0 0
 7 =   0 1 1 1
=================
Result  0 1 0 0
```

Where there is a '1' in both columns, is '1' is placed underneath in the result column, else place a zero. So the result is 0100 in binary, or 4 in decimal.

OR: The rule for OR is that where any bit is a '1' then a '1' is placed in the result column. e.g.

12 OR 7

```
12 = 1 1 0 0
 7 = 0 1 1 1
=============
Result 1 1 1 1
```

So the result is 15 decimal.

XOR: The rules are that if both bits are a '1' then the result is a '0', but if only one of them is a '1' then the result is a '1' e.g.

12 XOR 7

```
12 = 1 1 0 0
 7 = 0 1 1 1
=============
Result 1 0 1 1
```

or 11 decimal. Lets do another one,

8 XOR 11

```
 8 = 1 0 0 0
11 = 1 0 1 1
=============
Result 0 0 1 1
```

or 3 decimal.

NOT: The not command works on a single number, and you invert the bit pattern, i.e. where there was a '1' you make it a '0', and where there was a '0', you change it to a '1'. e.g.

NOT 10

```
      10 = 1 0 1 0
=================
Result 0 1 0 1
```

so it becomes 5.

The above commands are useful for machine programming, sprite collision detection routines, memory modification utilities and disassemblers. The average programmer will not use them very much, but they could be used a great deal with the disk drive unit.

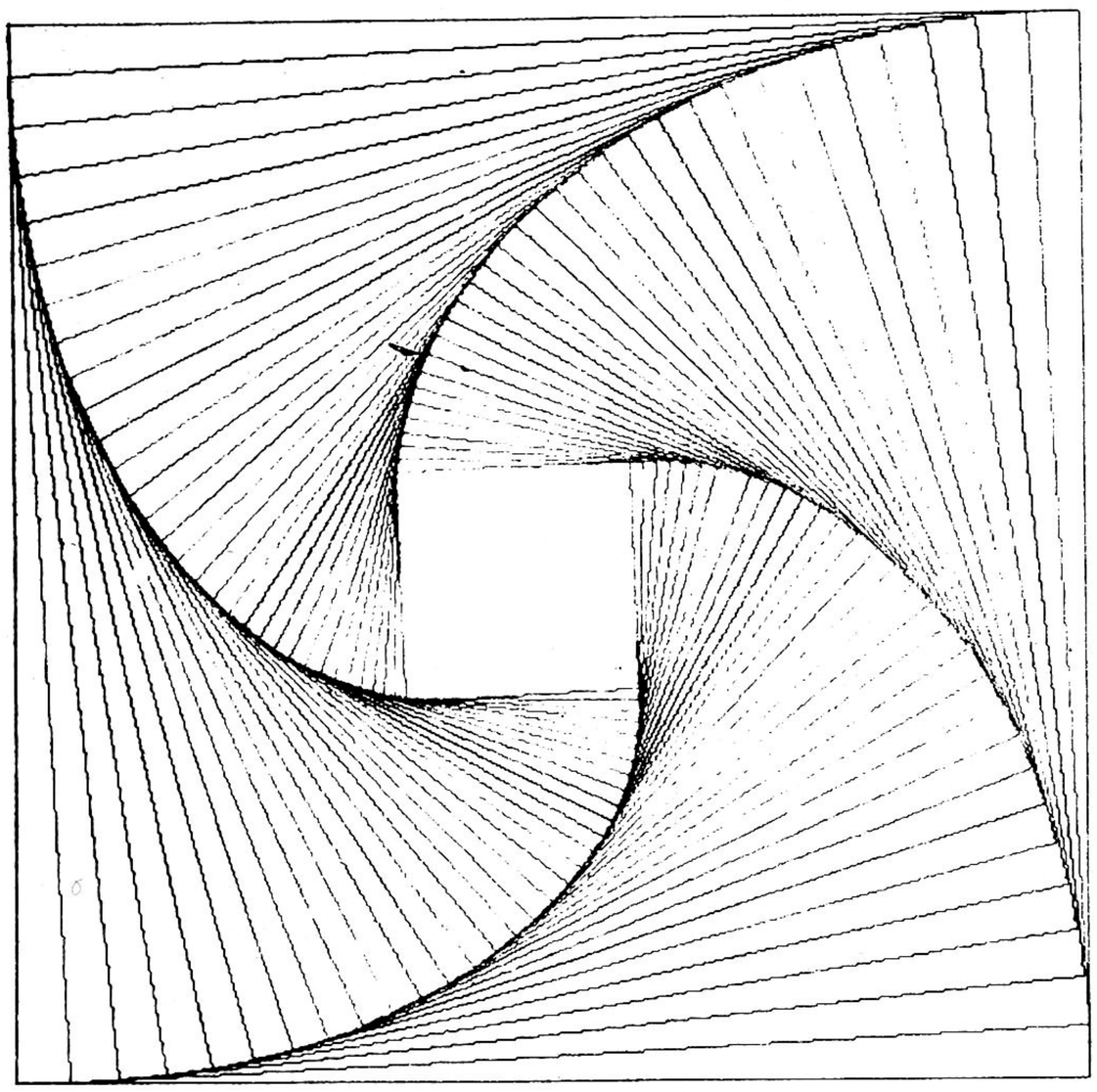
CALL: The call statement is used to transfer control from a basic program to a machine language program. The format of this command is

CALL address

where address is preferably a value in hexidecimal within the limits of &HOOOO to &HFFFF. The machine code program may be either a program which has been poked into memory beforehand, or may be a ROM routine, which is part of the basic language system. The following program illustrates the setting up of a machine language routine starting at &HAOOO, and the CALL &HAOOO is used to execute that routine.

The program loads the 'A' register with the cursor code, the 'B' register with the number of times to print it, then calls the PRINT routine in the internal basic ROM which prints the cursor thirty-two times at the top of the screen. The &HC9, the last data statement indicates that the machine language program is ended, and informs the processor to return to basic.

# SPIRAL



**"Try this"**

## PRINTER DEMONSTRATION

```
10 SCREEN 2,2:COLOR2,1,,1:CLS:POSITION
 (128,96)
20 X=0:Y=0:A=0:C=5
30 FOR R=2 TO 50 STEP 5
40 GOSUB 1000
50 NEXT R
60 LINE (X,Y)-(X,-96),5
70 X=-1:Y=2:A=180:C=9
80 FOR R=2 TO 50 STEP 5
90 GOSUB 1000
100 NEXT R
105 LINE(X,Y)-(X,95),9
108 SCREEN2,2
110 PAINT (-128,-96),5
115 COLOR 2,9,(-128,-96)-(127,95)
120 GOTO120
1000 FOR A=A TO A+180 STEP 30
1010 X2=X:Y2=Y
1020 X=X+SIN(RAD(A))*R
1030 Y=Y+COS(RAD(A))*R
1040 LINE (X2,Y2)-(X,Y),C
1050 NEXT A
1060 IF A=180 THEN A=0 ELSE A=0
1070 RETURN
```

```
10 REM *** DEMO 1 SQUARES ***
20 X1=  0:Y1=0
30 X2=  0:Y2=-480
40 X3=480:Y3=-480
50 X4=480:Y4=0
60 LPRINT CHR$(18)
70 REM main loop
80 A$="M"+STR$(X1)+","+STR$(Y1)
90 LPRINT A$
100 A$="D"+STR$(X2)+","+STR$(Y2)+","+S
TR$(X3)+","+STR$(Y3)
110 B$="D"+STR$(X4)+","+STR$(Y4)+","+S
TR$(X1)+","+STR$(Y1)
120 LPRINT A$
130 LPRINT B$
140 P1=X1:Q1=Y1
150 X1=INT(X1+(X2-X1)/20):Y1=INT(Y1+(Y
2-Y1)/20)
160 X2=INT(X2+(X3-X2)/20):Y2=INT(Y2+(Y
3-Y2)/20)
170 X3=INT(X3+(X4-X3)/20):Y3=INT(Y3+(Y
4-Y3)/20)
180 X4=INT(X4+(P1-X4)/20):Y4=INT(Y4+(Q
1-Y4)/20)
190 IF N=31 GOTO220
200 N=N+1
210 GOTO70
220 LPRINT "A"
230 END
```

# CANDY KID

**This is a basic**

## Maze Chase Game

**BY SEGA**

```
10 REM *** CANDY KID ***
20 MAG1:GOSUB 700:GOSUB 810
30 HS=0
40 SC=0:CK=3:ST=1:SS=1
50 IF CK<1 THEN GOTO 1000
60 CX=10:CY=5:BX=170:BY=165:CA=0
70 SCREEN 2,2:CLS:GOSUB 310
79 REM --- MAIN RUTINE ---
80 A$=INKEY$
90 IF A$=CHR$(28)THEN IF CX<170 THEN C
X=CX+16
100 IF A$=CHR$(29)THEN IF CX>10 THEN C
X=CX-16
110 IF A$=CHR$(30)THEN IF CY>5 THEN CY
=CY-16
120 IF A$=CHR$(31)THEN IF CY<165 THEN
CY=CY+16
130 UY=CY+4:A=UPEEK(INT(UY/8)*256+INT(
CX/8)*8+UYMOD8)
140 SPRITE3,(CX,CY),0,2:SPRITE2,(CX,CY
),4,11
150 IF A=3 THEN BEEP:CA=CA+1:SC=SC+10:
BLINE(CX,CY)-(CX+15,CY+15),15,BF:GOSUB
 270
160 IF A=19 THEN CK=CK-1:GOTO 280
170 IF BX>CX THEN BX=BX-8:GOTO 200
180 IF BX<CX THEN BX=BX+8:GOTO 200
190 BX=CX
200 IF BY>CY THEN BY=BY-8:GOTO 230
210 IF BY<CY THEN BY=BY+8:GOTO 230
220 BY=CY
230 SPRITE1,(BX,BY),8,5:SPRITE0,(BX,BY
),12,6
240 IF BX>CX-12 AND BX<CX+12 AND BY>CY
-12AND BY<CY+12 THEN CK=CK-1:GOTO 280
250 IF CA=CS THEN GOTO 290
260 GOTO 80
269 REM ===== SCORE =====
270 BLINE(195,50)-(250,60),15,BF:CURSO
R195,50:COLOR 1:PRINTSC:RETURN
279 REM **** KID DEAD ***
280 GOSUB 630:CURSOR200,120:COLOR1:PRI
NT"SPLAT !!":FOR I=0 TO 1000:NEXT I:GO
SUB 50
289 REM *** STAGE DISPLAY ***
290 GOSUB 560:SC=SC+ST*300:GOSUB270:CU
RSOR200,120:PRINT"NICE !!"
300 FOR I=0 TO 1000:NEXT I:ST=ST+1:SS=
SS+1:GOTO 50
309 REM ====== GAME DISPLAY ======
310 LINE(7,2)-(191,185),1,B:FOR X=14 T
O 174 STEP 16:FOR Y=9 TO 169 STEP 16
320 C=INT(RND(1)*12)+2:COLORC:CURSORX,
Y:PRINT"#":NEXT Y:NEXT X
330 IF SS=4 THEN SS=1
340 IF SS=1 THEN CS=96:GOTO 370
350 IF SS=2 THEN CS=98:GOTO 390
360 IF SS=3 THEN CS=85:GOTO 420
370 FOR X=26 TO 154 STEP 32:FOR Y=21 T
O 149 STEP 32:GOSUB 490:NEXT Y:NEXT X
380 GOSUB 500:GOSUB 560:RETURN
390 FOR X=26 TO 154 STEP 64;FOR Y=21 T
O 149 STEP 32:GOSUB 490:NEXT Y:NEXT X
400 FOR X=58 TO 122 STEP 64:FOR Y=37 T
O 122 STEP 32:GOSUB 490:NEXT Y:NEXT X
410 GOSUB 500:GOSUB 560:RETURN
420 RESORE 780
430 READ X
440 READ Y
450 IF Y=-1 THEN GOTO 430
460 IF Y=-2 THEN GOTO 480
470 GOSUB 490:GOTO 440
480 GOSUB 500:GOSUB 560:RETURN
490 COLOR 1:CURSORX,Y:PRINT"$" :CURSOR
 X,Y+8:PRINT"x":CURSORX+8,Y:PRINT"&":C
URSOR X+8,Y+8:PRINT "'":RETURN
500 CURSOR 195,15:COLOR9:PRINT"HI-SCOR
E":CURSOR 195,25:PRINT HS
510 CURSOR 195,40:COLOR1:PRINT"SCORE":
CURSOR 195,50:PRINT SC
520 CURSOR 195,65:COLOR2:PRINT"KID":CU
RSOR 195,75:PRINT CK
530 CURSOR 195,90:COLOR5:PRINT"STAGE":
CURSOR 195,100:PRINT ST
540 FOR C=1 TO 13:COLOR C:CURSOR 200,1
50:PRINT"*******":CURSOR 200,158:PRINT
"*CANDY*":CURSOR 200,166:PRINT"* KID *
":CURSOR 200,174:PRINT"*******":NEXT C

550 RETURN
559 REM ==== SOUND  1 ========
560 RESTORE 790
570 READ T1
580 READ S1
590 IF S1=-1 THEN GOTO 570
600 IF S1=-2 THEN SOUND 0:RETURN
610 SOUND1,S1,15:FORI=0 TO T1:NEXT I
620 GOTO 580
629 REM ==== SOUND  2 =========
630 RESTORE 800
640 READ T2
650 READ S2
660 IF S2=-1 THEN GOTO 640
670 IF S2=-2 THEN RETURN
680 SOUND1,S2,15:FOR I=0 TO T2:NEXT I:
SOUND 0
690 GOTO 650
699 REM ===== CHR DATA READ =====
700 FOR I=0 TO 15:READ Q$:PATTERNS#I,Q
$:NEXT I
```

```
710 FOR I=35 TO 39:READ Q$:PATTERNC#I,
Q$:NEXT I:RETURN
720 DATA 030F1F3F33616173,7F33381C8F53
A050,C0F0F8FCCC8686CE,FECC1C38F1CA050A


730 DATA 0000000000C0C00,000C07030080
4020,0000000000303000,0030E0C000010204

740 DATA 03070F1F193070F9,FE9C0F0F0F3F
7F73,C0E0F0F8980C0E9F,7F39F0F0F0FCFECE

750 DATA 0000000000060600,000310101000
0000,0000000000606000,00C00808080000000

760 DATA C0C03C3C3C3C0303
770 DATA 100F45E543424161,62524A444444
040F,08F0A0A0E0408080,444E4424243820F0



779 REM ==== SCREEN DATA =====
780 DATA 26,21,37,53,85,117,133,149,-1
,42,21,149,-1,58,21,53,69,101,117,149,
-1,74,53,117,-1,90,21,149,-1,106,53,11
7,-1,122,21,53,69,101,117,149,-1,138,2
1,149,-1,154,21,37,53,85,117,133,149,-
2
789 REM ===== SOUND DATA 1 =====
790 DATA 20,523,659,784,659,784,659,-1
,60,784,-2
799 REM ===== SOUND DATA 2 =====
800 DATA 40,523,-1,20,523,523,-1,40,52
3,-1,20,622,587,587,523,523,494,-1,60,
523,-2
809 REM ==== TITAL =====
810 SCREEN 2,2:CLS:COLOR 8:CLS
820 CURSOR0,20:PRINT"      CANDY KID"
900 CURSOR 58,108:COLOR 2:PRINT ".....
...CANDY KID"
910 CURSOR 58,124:COLOR 5:PRINT ".....
...CANDY MAMA"
920 X=42:Y=137:COLOR1:CURSOR X,Y:PRINT
"$":CURSORX,Y+8:PRINT"%":CURSORX+8,Y:P
RINT"&":CURSORX+8,Y+8:PRINT"'"
930 CURSOR 58,140:PRINT".....BAC
TERIAL"

940 CURSOR 58,156:PRINT".....CAN
DY 10P.T.S"
950 FORX1=0 TO 42 STEP2:SPRITE 3,(X1,1
01),0,2:SPRITE2,(X1,101),4,11:SPRITE1,
(X1,117),8,5:SPRITE0,(X1,107),12,6:NEX
TX1
960 FOR C=2 TO 13:COLOR C:CURSOR46,157
:PRINT"#":BEEP:FOR I=0 TO 50:NEXT I:NE
XT C
970 GOSUB 560:CURSOR60,180:COLOR1:PRIN
T"Please push [ CR ] key"
980 IF INKEY$=CHR$(13) THEN RETURN
990 GOTO 980
999 REM ==== GAME OVER ====


1000 CLS:IF SC>HS THEN HS=SC
1010 CURSOR 100,40:COLOR 9:PRINT "GAME
 OVER"
1020 CURSOR 90,70:COLOR 13:PRINT "HI-S
CORE";HS

1030 CURSOR 90,90:COLOR1:PRINT"YOUR SC
ORE";SC
1040 CURSOR50,120:COLOR 2:PRINT"Play a
gain?  - push [ CR ] key"
1050 CURSOR 50,140:COLOR 5:PRINT"End
        - push [ E ] key"

1060 IF INKEY$=CHR$(13) THEN GOTO 40
1070 IF INKEY$="E" THEN END
1080 GOTO 1060
```


**Introducing**

for the SEGA SC 3000

1

THE SPIDERSOFT

Early Learner Series

for Ages 4 to 8 By Grant Anderson

**THE SPIDERSOFT EARLY LEARNER SERIES**

This series of six programmes is produced by a N.Z. teacher for the pre-school to early primary area. Entertaining games combined with concrete cues encourage children to practise important skills.

| Titles | Age Range | Titles | Age Range |
|---|---|---|---|
| 1 Comparing and arranging shapes | 3 to 5 | 2 Numeral recognition | 4 to 5 |
| 3 Addition + subtraction level 1 | 5 to 7 | 4 Addition + subtraction level 2 | 6 to 8 |
| 5 Adding suffixes (to words) | 5 to 8 | 6 Alphabet + keyboard skills | 5 to 8 |

Price: $18.95 each or any 2 programs for $29.99
Special offer: The complete series (6 programmes) for $64.95.

**SPIDERSOFT "TIMES TABLES"**
**It tests, it insists on corrections, then it rewards with the exciting maze game "Stealth". Persistence brings increasing rewards.**
**For age 8 upwards** **Price $19.95**

**ORDER FORM** RETAILER ENQUIRIES ARE WELCOME.

**To SUPERCOM DATA SERVICES LTD., Box 34-313, Birkenhead, Auckland.**

**Tick the titles you wish to order**

**1** ☐ **Name**..... **2** ☐ **Address**.....

**3** ☐ ..... **4** ☐ **Bankcard or Visa No. (if appropriate)**

**5** ☐ ..... **6** ☐ **Expiry Date**.....**Signature**

**Tables** ☐ **Total Price**..... .....

D1

# GRAPHICS DEMONSTRATIONS

The following 8 programs are small graphics demonstrations showing just what you can do with line and circle statements on your Sega. Each program produces its own unique pattern that can be changed by altering some of the variables giving some startling results.

```
10 SCREEN 2,2:CLS
20 FOR N=3 TO 12
30 IF 360/N=INT(360/N) THEN GOSUB 100
40 NEXT N
50 GOTO 20
100 CLS
110 FOR R=90 TO 10 STEP -10
120 ZX=R+128:ZY=90
140 PSET(ZX,ZY),1
150 FOR TH=0 TO 360 STEP 360/N
160 GOSUB 300
180 LINE -(X,Y),1
190 NEXT TH,R
195 GOSUB 500
200 RETURN
300 X=R*COS(RAD(TH))+128
320 Y=R*SIN(RAD(TH))+90
340 RETURN
500 R=90:FOR TH=0 TO 360 STEP 360/N
510 GOSUB 300
520 LINE (128,90)-(X,Y),1
530 NEXT TH:RETURN
```

```
10 SCREEN 2,2:CLS
20 R=45
30 FOR I=0 TO 2*PI STEP PI/4/R
40 Y=R*SIN(I):PSET(X,Y+95):PSET(X+R*1.
5,Y+95):PSET(X,2*Y+95):PSET(X+R*1.5,2*
Y+95)
50 X=X+0.5:NEXT I
60 FOR J=0 TO 2*PI STEP PI/R*2
70 YY=R*SIN(J):LINE(XX,2*YY+95)-(XX+R*
1.5,2*YY+95)
80 LINE (XX,YY+95)-(XX+R*1.5,YY+95)
90 XX=XX+4:NEXT J
100 GOTO 100
```

```
10 SCREEN 2,2:CLS
20 R=6:P=6:N=8:L=R*P:K=2*PI/N:C=0
30 FOR F=0 TO L STEP L
40 FOR G=1 TO 191/(L*2)
50 FOR H=1 TO 255/(L*2)
60 CL=CL+1
70 FOR I=1 TO N
80 FOR J=0 TO P
90 A=R*J*COS(K*I)+L*2*H+F-L:B=R*J*SIN(
K*I)+2*L*G+F-L
100 C=R*(P-J)*COS(K*(I+1))+L*2*H+F-L:D
=R*(P-J)*SIN(K*(I+1))+L*2*G+F-L
110 LINE (A,B)-(C,D),CL
120 NEXT J,I,H,G,F
130 GOTO 130
```

```
10 SCREEN 2,2:CLS
20 R=90:M=3:N=30:K=2*PI/M
30 FOR I=0 TO K STEP K/N
40 LINE (R*COS(I)+122,R*SIN(I)+95)-(R*
COS(I+K)+122,R*SIN(I+K)+95)
50 LINE -(R*COS(I+2*K)+122,R*SIN(I+2*K
)+95)
60 LINE -(R*COS(I)+122,R*SIN(I)+95)
70 NEXT I
80 GOTO 80
```

```
10 SCREEN 2,2:CLS
20 L=40:A=147:B=190
30 FOR I=1 TO 13
40 FOR J=1 TO I+2
50 K=2*PI-J*PI*(1-I/(I+2)):AA=A+L*COS(
K):BB=B+L*SIN(K)
60 LINE (A,B)-(AA,BB)
70 A=AA:B=BB:NEXT J,I
80 GOTO 80
```

```
10 SCREEN 2,2:CLS
20 FOR I=0 TO 255 STEP 4
30 LINE(I,0)-(255-I,191),1:NEXT I
40 FOR J=191 TO 0 STEP -3
50 LINE(255,191-J)-(0,J),1:NEXT J
55 C=C+1:IF C=15 THEN C=1
60 COLOR ,C,(0,0)-(255,191),C
70 GOTO 55
```

```
10 SCREEN 2,2:CLS:C=1
20 R=20:A=0
30 FOR K=R TO 191-2*R STEP 2*R
40 FOR I=R*2 TO 255 STEP R
50 CIRCLE (I-R,K+A),R,C,1,0.5+0.5/3,1
60 CIRCLE (I,K+A),R,C,1,0.5/3,0.5
70 NEXT I:C=C+1:IF C=15 THEN C=1
80 A=A+10:NEXT K
90 GOTO 20
```

```
10 SCREEN 2,2:CLS
20 N=5:D=8:K=2*3.14/N:R=90:A=0:KD=K/D:
C=1
30 FOR I=1 TO N
40 LINE (R*COS(A)+122,R*SIN(A)+95)-(R*
COS(A+K)+122,R*SIN(A+K)+95),C
50 A=A+K:NEXT I
60 R=R/(COS(KD)+SIN(KD)/TAN((3.14-K)/2
)):A=A+KD
70 IF R>47 THEN 30
80 C=C+1:IF C=15 THEN C=1
90 GOTO 20
```

# How to Program in Machine Code Language on The SEGA SC3000

by COLIN SMITH

The following article is a brief introduction to programming in machine code.

When a program is entered in Basic, what the computer must first do, is convert the commands from Basic into its own machine language, and refer to its own memory locations to carry out the routines which you are telling it to carry out.

Programming in machine code, is effectively giving the computer the addresses which hold the routines in its own language therefore cutting out one stage of the process which means operations are speeded up considerably.

Understanding machine language may initially seem very confusing, and the first few articles will require a certain amount of patience, and acceptance of some pieces of information, which may seem to be irrelevant, and in same cases seem very difficult to understand.

However in time all will become clear, and by referring back to the early articles the statements which are made will be explained, and you will be able to perform the tasks which you are now having to write in basic, in machine code, to greatly improve the performance of the computer.

What is machine code language?

It is the language that a computer uses to talk to itself and other outside devices. This applies to million dollar time-sharing installations, right down to the humble pocket calculator. When using languages such as Basic or Pascal, these are broken down or interpreted into machine code segments, which are understood and operated on by the computer. A machine code program will run faster and it is possible to predict how long a program will take to run. It is possible to pack information into less space and there are things which a higher level language cannot achieve, such as peripheral control.

To start programing in machine code, as you are going to directly manipulate the computer there are a few things going on beneath the keyboard that you will need to understand. This issue will give you information to refer back to as you continue through this series.

The computer chip used in the Saga SC3000 is called a Z80, it is a popular chip, used in many applications. There are several control functions you will encounter which will alter the way in which the Z80 will operate, some of these you have no control over but will need to know.

### 1/Clock:

Not a time telling version, but a crystal which organises the Z80 to operate in an orderly fashion. This clock runs at a speed of 3.58 Megahertz or it turns on and off at 3.58 million times a second. Each on/off period is called a T state, so the time taken for 1 T state is 0.00000028 of a second. Each instruction step can take from 3 to 6 T states and there can be several steps in an operation, so if we know what operations are taking place we can accurately time them.

### 2/Reset Line:

When a signal is sent to the reset line this makes the computer go to the very first, address in memory which is ØØØØH. (Note: your reset button is 'not' connected to this line).

### 3/Wait Line:

Not all devices (electronic or otherwise) are as fast as the Z80. When these devices are called on to operate they send a signal causing the computer to stop and wait until they are ready, this will slow any programe down.

### 4/Interupt Line:

There are devices which operate independently of the computer and there are occasions when it is necessary for these devices to stop the computer from one programe and make it run another. It is possible to stop this from happening, this is called 'Masking the Interupt.'

### 5/Non-Maskable Interupt:

As it implies, you cannot stop this signal from interupting the computer. It is this line that your reset button is fastened too. When your computer is first switched ON it has rubbish in its internal memories and would start operating on incorrect data. What is required is a way to start the computer in an orderly fashion. This is achieved by deliberately signaling the Reset Line so that it will fetch its first instruction from ØØØØH. This is known as a cold start and a cold start clears everything from memory. To save your programes being lost, the reset key is connected to the N.M.I. Line, making the computer jump to a programe in R.O.M. allowing you to escape from a programe without having your data removed. This is called a 'warm start.'

## INSIDE THE Z80 CHIP

The chip has 26 bytes of read/write memory that is accessable to the programer. These are arranged into 8 and 16 bit registers.

### 1/Stack Pointer(SP):

The stack is an area of memory used for temporary storage of data. The stack pointer holds the 16 bit address of the current location of the stack. It is arranged as a LIFO file, (last in first out).

### 2/Program Counter (PC):

The program counter holds the 16 bit address of the current instruction that is being accessed in memory. The program counter is automatically increased by 1 count (Incremented) after each operation. When a program jumps to a new location, the location is placed directly into the program counter.

### 3/Index Registers (IX & IY):

These are two 16 bit registers that are used as 'Base' values from which to access information.

### 4/Interupt Register (I):

When operated in a certain 'mode' the computer will jump to a new location when interupted. The new locations address is stored in the Interupt Register (High Byte) with the interupting device supplying the low byte address.

### 5/Memory Refresh Register (R):

When the Z80 is used with 'Dynamic Memory' the Z80 supplies the memory refresh signal as well as the address which is held in the refresh register. This operation is completely transparent to the operator and has no effect on the speed of any program.

### 6/Accumulator (A) and Flag (F) Registers:

The accumulator holds the result of any arithmetic or logical operation. The flag register holds the result of any special condition that applies to the result in (A).

There are two sets of Accumulators and Flags and only one set can be used at a time.

MAIN REGISTER SET

| Accumulator | Flag |
|---|---|
| A | F |
| B | C |
| D | E |
| H | L |

ALTERNATE REGISTER SET

| Accumulator | Flag |
|---|---|
| A' | F' |
| B' | C' |
| D' | E' |
| H' | L' |

### 7/General Purpose Registers:

There are left, 12 registers, these are arranged into pairs as 16 bit registers, although they can be used as individual 8 bit registers. The register pairs are called BC,DE, and HL plus BC' DE' and HL'. BC and BC' are alternate sets, the same as DE:DE' HL:HL' and only one alternate set can be used at a time.

SPECIAL PURPOSE REGISTERS

| | |
|---|---|
| Interupt Vector 1 | Memory Refresh R |
| Index Register | IX |
| Index Register | IY |
| Stack Pointer | SP |
| Program Counter | PC |

These articles to be continued next issue.

# PROGRAM OF THE MONTH

**Each issue, we would like to invite Sega owners to send in their programs for evaluation and possible entry into the Magazine. Not only games programs, but utility programs, and machine code or basic routines. Each program will be judged on originality, usefullness, or play value, graphic content or design and layout, individually depending on its use or function.**

**Any programs which are considered to be commercially viable, separate contact will be made to agree terms. All other programs accepted for inclusion into the magazine, will receive a cassette program of the writers choice.**

**One program each issue will be singled out as being the ''Star Program of the Month'', entitling the writer to select any one of Sega's game cartridges as their prize.**

# SEGA SUPERSTAR

## HIGH SCORE TABLE

A new regular feature beginning in your next issue of the Sega User Magazine, will be a top 5 score table for all Sega game cartridges.

The player recording the highest points tally each month will receive a free cassette program, and have their name published in the Magazine, along with their score and the scores of the four runners up.

To have your score entered, we must have some form of verification as to its validity, ideally a photograph of the screen, showing the recorded high score. In games where the high score may be flashing, the screen action can be frozen, by depressing the Reset key.

To get you started, here are just a few high scores to aim for.

| | | | |
|---|---|---|---|
| Star Jacker | 380,000 | Pop Flamer | 71,600 |
| Monaco GP | 122,150 | Video Flipper | 110,600 |
| | Pacar | 1,526,400 | |

# GUNSLINGER

**BY GLENN HOWARD OF INVERCARGILL**

```
5 GOTO10
6 CLS:PRINT"Bye Bye"
7 FORT=1TO500:NEXT
8 END
10 SCREEN2,2:CLS:M=1:N=13:SE=20:NUM=10
0
20 COLOR15,1,(0,0)-(255,191),1
22 FORCO=2TO15
24 COLORCO
30 CURSOR68,63:PRINTCHR$(17);"GUNSLING
ER"
35 PRINT,,CHR$(16);"                By Gl
enn Howard"
37 NEXT
40 PRINT,,,,"          Press any key to
 begin"
60 S$=INKEY$
70 IFS$=""THEN60
110 CLS:PRINT"Do you want instructions
? (Y/N)"
140 G$=INKEY$
150 IFG$=""THEN140
160 IFG$="Y"ORG$="y"THEN1000
170 IFG$="N"ORG$="n"THEN1700
180 GOTO140
1000 SCREEN1,1:CLS:COLOR15,1:CURSOR13,
1:PRINT"INSTRUCTIONS"
1010 CURSOR12,2:PRINT"
1020 PRINT" In this game there are two
 men that  walk towards each other. Ea
ch takes 15steps and then stops. When
they stop  walking the word ""DRAW"" a
ppears on thescreen. This is an instru
ction to you to fire. You can do so by
 pressing the""F"" key."
1030 PRINT,," A word of warning. If yo
u try to firebefore the word ""DRAW""
appears then  the man will automatica
lly shoot you. You are the player on t
he left."
1040 PRINT,,,,"      Press any key to
continue"
1100 RI$=INKEY$
1110 IFRI$=""THEN1100
1120 GOTO1700
1700 SCREEN2,2:CLS:C=6:GOTO1730
1730 COLOR6,3,(0,128)-(255,191),14
1740 COLOR6,15,(0,0)-(255,134)
1750 CURSOR145,73:PRINT"o"
1760 CURSOR170,73:PRINT"o"
1770 CURSOR143,81:PRINT"SALOON"
1780 CURSOR110,62:PRINT"

1790 CURSOR110,67:PRINT"

1800 CURSOR125,28:PRINT"NOTHIN GULCH"
1810 RESTORE10180
1820 FORI=1TO41
1830 READX1,Y1,X2,Y2
1840 LINE(X1,Y1)-(X2,Y2),C,B
1845 NEXT
1850 FORI=1TO 18
1860 READX1,Y1,X2,Y2
1865 LINE(X1,Y1)-(X2,Y2),C
1867 NEXT
1870 FORI=41TO48:LINE(155,I)-(165,I),6
:NEXT
1871 PAINT(160,17),6:PAINT(114,34),6:P
AINT(200,34),6
1875 C=2:PSET(35,135),C
1880 FORI=1TO22
1885 READX1,Y1
1890 LINE-(X1,Y1),C
1895 NEXT
1900 PAINT(37,134),2:GOTO2010
2010 MAG3
2015 TIME$="00:00:00"
2020 F=0:Y=103:X=220:G=F:D=101:NUM=10
2030 GOTO2050
2040 COLOR6,15,(0,0)-(255,127),14
2050 PATTERNS#1,"0000000000000000"
2060 PATTERNS#20,"0707070F05090702"
2070 PATTERNS#21,"0F0F3F0F0F050505"
2080 PATTERNS#22,"0000008000000000"
2090 PATTERNS#23,"00c0400000000000"
2100 PATTERNS#4,"0707070F05040702"
2110 PATTERNS#5,"0F3F2F4F0F0A0A05"
2120 PATTERNS#6,"0000008000800000"
2130 PATTERNS#7,"0000C00000000000"
2140 PATTERNS#28,"0707070F05090702"
2150 PATTERNS#29,"0F0F3F0F0F05050A"
2160 PATTERNS#30,"0000008000000000"
2170 PATTERNS#31,"00C0200000000000"
2180 PATTERNS#36,"0707070F05040702"
2190 PATTERNS#37,"0F3F2F2F0F0A0A0A"
2200 PATTERNS#38,"0000008000800000"
2210 PATTERNS#39,"0000C00000000000"
2230 SPRITE1,(G,D),1,1
2240 SPRITE10,(X,Y),20,M
2250 SPRITE11,(F,Y),4,N
2260 SOUND1,190,15:X=X-4:F=F+4:FORI=1T
050:SOUND0:NEXTI
2270 IFX<162THEN3280
2280 GOTO4310
3280 CURSOR55,35:PRINTCHR$(17);"DRAW"
4200 CH$=INKEY$
4205 CH=CH+1
4207 IFCH>SE THEN4400
4210 IFCH$=""THENCH=CH+1
4220 IFCH$="f"ORCH$="F"THEN4222
4221 GOTO4200
4222 IFCH<3THEN4400:IFCH>SETHEN4400
```

```
4228 GOTO4300
4229 GOTO4220
4300 X=X+15:SOUND4,2,10:SOUND4,1,15:FO
RI=FTOXSTEP5:SPRITE1,(X,I),0,1:NEXTI:G
OTO5000
4305 X=X-15:SOUND0
4310 SPRITE10,(X,Y),28,M
4320 SPRITE11,(F,Y),36,N
4325 IFINKEY$<>""THEN6000
4330 GOTO2060
4400 SOUND5,0,15:FORO=XTOFSTEP-5:VPOKE
&H3B05,O:NEXTO:GOTO6000
5000 COLOR15:SE=SE-.2:NUM=NUM+3:CH=0:S
OUND0:TE=TE+1:SC=TE*NUM
5005 IFSE<8THENSE=8
5010 CURSOR55,35:PRINT"DRAW":COLOR2,15
:SCREEN1,1:CLS:COLOR2,1
5020 CURSOR1,1:PRINTCHR$(16);"SC-";SC
5030 CURSOR14,1:PRINT"SET-";TE
5032 CURSOR10,3:PRINT"TIME-";TIME$
5033 IFSC>HITHENHI=SC
5035 CURSOR25,1:PRINT"HI-";HI
5040 GOTO5200
5100 SCREEN2,2:BLINE(55,35)-(100,45),,
BF
5110 MAG3
5135 F=0:Y=103:X=220:G=F:D=101:CH=0
5140 GOTO2200
5200 FORTI=1TO500
5210 NEXT
5220 GOTO5100
6000 SOUND0:SCREEN1,1:CLS:COLOR15,1
6010 CURSOR10,3:PRINT"GAME OVER"
6012 CURSOR12,8:PRINTCHR$(16);"SC-";SC
:CURSOR12,10:PRINT"SET-";TE:CURSOR12,1
2:PRINT"HI-";HI
6013 CURSOR3,15:PRINT"TIME-";TIME$
6015 GOTO 20000
6020 FORGA=1TO15
6030 IFGA=75THEN6100
6040 NEXT
6100 PRINT:PRINT:PRINT"Play again? (Y/
N)"
6120 AN$=INKEY$
6130 IFAN$=""THEN6120
6140 IFAN$="Y"ORAN$="y"THEN6200
6150 IFAN$="N"ORAN$="n"THEN6
6160 GOTO6120
6200 TIME$="00:00:00":M=1:N=13:SE=15:C
H=0:TE=0:SC=0:F=0:Y=103:X=220:G=F:D=10
1:SCREEN2,2:BLINE(55,35)-(100,43),,BF:
GOTO2150
6300 SCREEN2,2:CLS
6310 GOTO1700
10180 DATA 120,74,200,128,118,128,202,
130,115,130,205,132,112,132,208,134,14
0,79,180,89,112,62,208,74,111,61,209,6
1,195,74,125,36,122,26,198,36,121,75,1
22,128,198,75,199,128,145,90,146,130,1
74,90,175,130
10190 DATA 145,99,160,121,175,99,160,1
21,147,97,148,123,172,97,173,123,151,1
02,157,118,163,102,169,118,163,104,169
,106,163,108,169,110,163,112,169,114,1
63,116,169,118,151,104,157,106,151,108
,157,110,151,112,157,114,151,116,157,1
18
10200 DATA 127,87,137,112,126,112,138,
113,127,92,137,97,127,102,137,107,183,
87,193,112,182,112,194,113,183,92,193,
97,193,102,183,107,132,40,144,58,131,5
8,145,59,154,40,166,58;153,58,167,59,1
76,40,188,58,175,58,189,59,122,83,130,
75
10210 DATA 122,82,129,75,198,83,190,75
,198,82,191,75,132,87,132,112,188,87,1
88,112,111,36,122,36,109,36,122,25,122
,25,160,16,160,16,198,26,198,26,210,36
,210,36,195,36,138,40,138,58,133,49,14
3,49,160,40,160,58,155,49,165,49,177,4
9,187,49,182,40,182,58
10220 DATA 35,110,20,110,20,85,27,85,2
7,102,35,102,35,45,45,45,45,70,52,70,5
2,50,60,50,60,77,45,77,45,110,52,110,5
2,95,60,95,60,117,45,117,45,135,35,135

20000 IFTE<5THENPRINT:PRINT"   RATING
 PATHETIC":GOTO6020
20010 IFTE<10THENPRINT:PRINT"   RATING
  CRETIN":GOTO6020
20020 IFTE<15THENPRINT:PRINT"   RATING
  COWBOY":GOTO6020
20030 IFTE<20THENPRINT:PRINT"   RATING
  DEPUTY":GOTO6020
20040 IFTE<35THENPRINT:PRINT"   RATING
  GUNSLINGER":GOTO6020
20050 IFTE>35THENPRINT:PRINT"   RATING
                MARSHAL":GOTO6020
```


**COMPUTER DEMONSTRATORS REQUIRED**

**Grandstand Leisure require keen friendly outgoing people to work full-time for instore sales and demonstration of computers and electronic games. Positions are available nationally.**

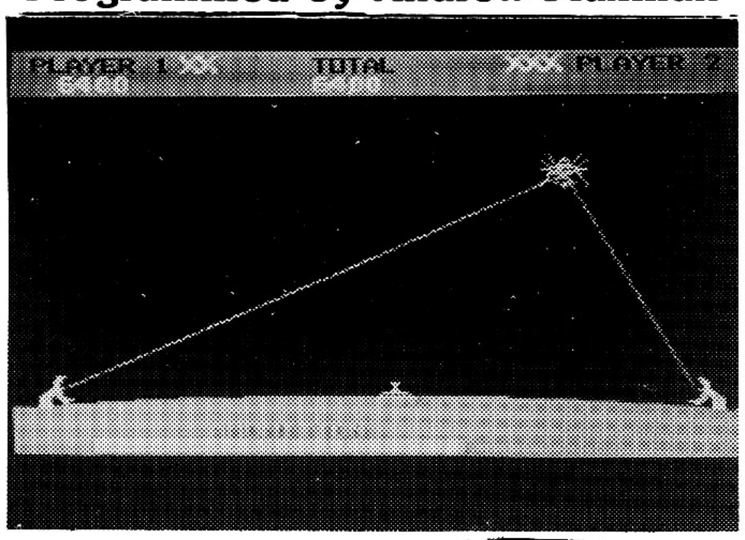
**Applicants should contact:**

**Phil Kenyon**
**Les Kenyon**

**On Auckland 504-033** **Urgently**

# LASER BLAST

**Programmed by Andrew Flaxman**




Your Mission: To protect your planet from invasion. You have 3 weapons:

A planet based laser tank
A space laser
and 3 smart bombs

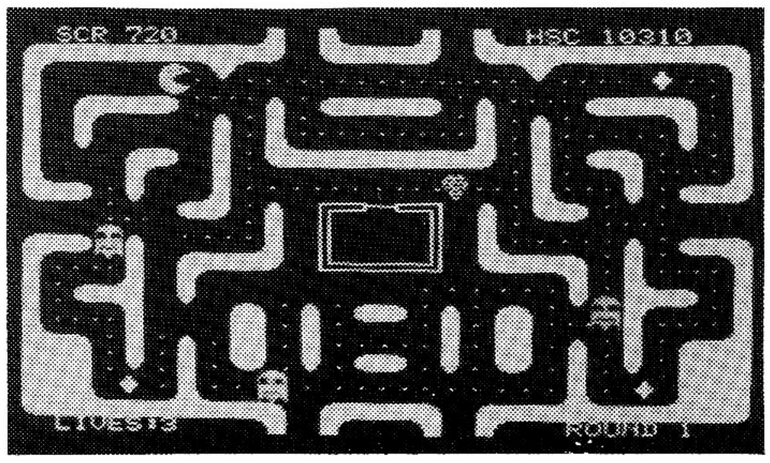
This one or two player game has some very clever features. You can warp off one side screen and on to the other. The sprites used for the men and the invaders are superb. When you are invaded an alien waddles around the screen blowing up your base and your weapons.

This game is a lot of fun for two players as your computer records your scores separately.

You are not only competing against the computer but against each other.

# MUNCHMAN

**Program by Terry Johnson**

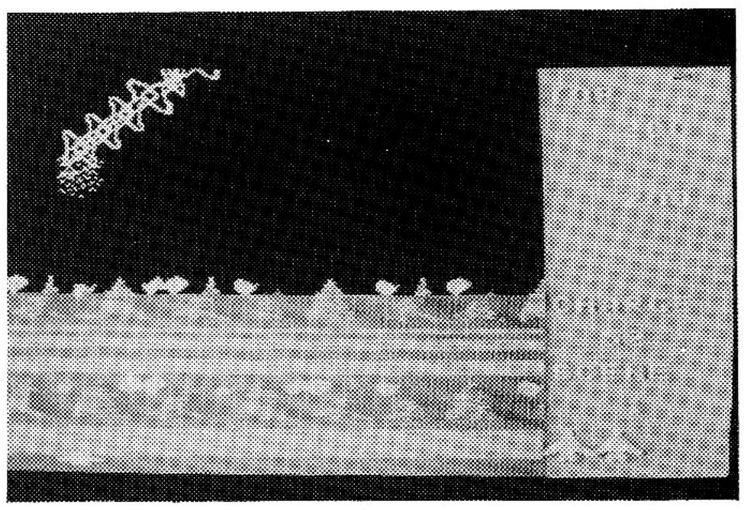



This is the classic maze chase game at its best. You move around the maze while eating up all the dots using either the joystick or the keyboard cursor keys for control. You also have four ghosts chasing you, with a very deadly mission. However revenge is sweet as you can attack these ghosts for a short period of time after you have eaten any one of the four "power pills" situated in the corners of the maze. Also keep a sharp lookout for the pieces of fruit dropped around the maze by the ghosts as these are worth a large bonus if you are skilled enough to eat them.

This game has to be seen to be believed. This is a machine coded cassette program therefore it is an extremely fast playing game with very good arcade quality graphics and sound.

# SOFTWARE REVIEW

# Exerion

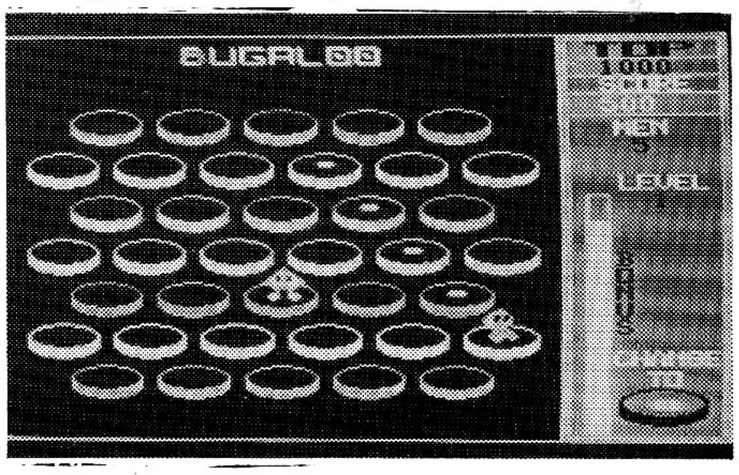


This incredibly realistic all action flight control game, puts you in charge of a fighter plane, that is the sole defender against hordes of attacking aliens. The sensation of speed achieved by the way the 3D scenerio changes below and on the horizen, coupled with the soaring movements of your aircraft have been captured with an incredibly accurate enthralling conversion from the arcade version which is currently thrilling arcade fans throughout the world.

Game players of all ages are sure to put this game in their library of all time favourites.

# BUGALOO

**Programmed By Andrew Flaxman**

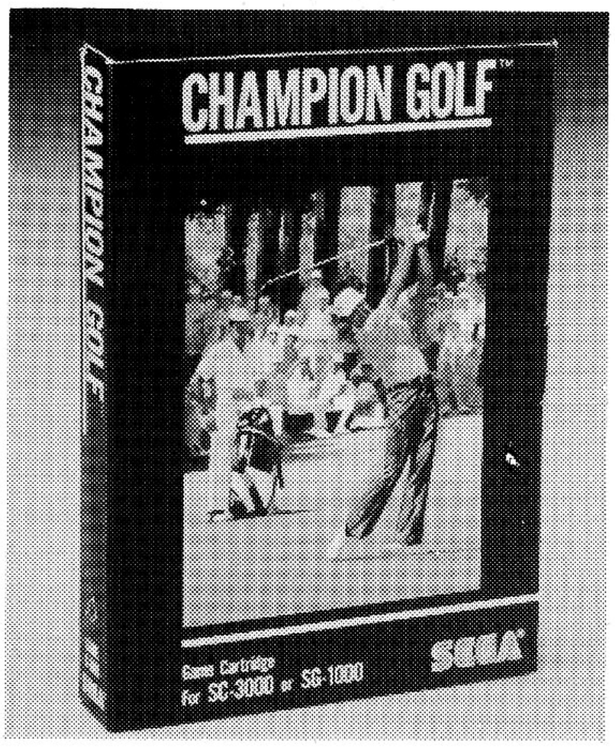



You were the unfortunate victim of circumstances as you were taken captive by the evil Gonzo race of aliens. They forced you to play their planet's survival game suspended in space.

You have to outsmart the Gonzos by jumping from platform to platform, leaving a trail of spots. When you have visited every platform, you have to start again, but it will be much harder. Every now and again a disk will appear and if you jump on it, 500 bonus points are awarded.

This one player game uses the SC 3000's graphic and sound capabilities to their fullest extent.

# Champion Golf Cartridge Review

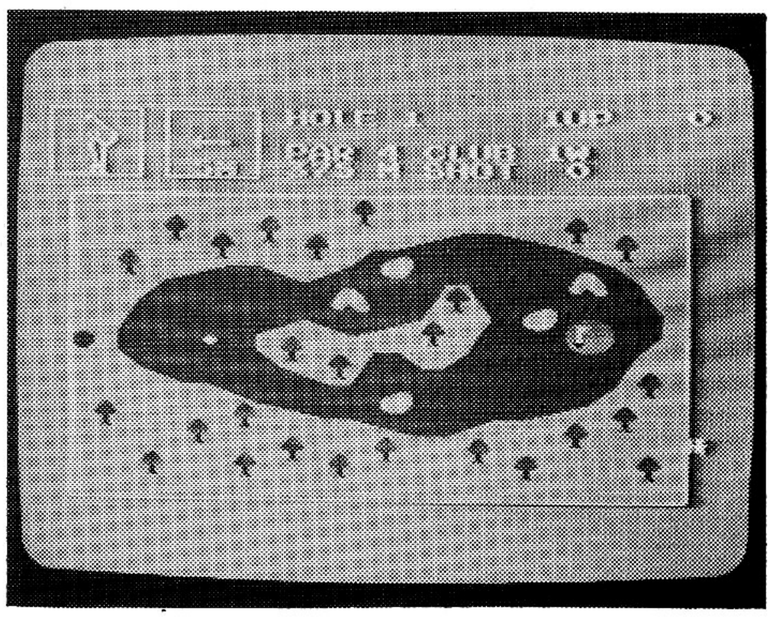



This is probably the most realistic golf game ever written for a computer. The objective of this one or two player game is to get around all nine holes of the very demanding golf course. This is not as easy as it sounds as you have many obstacles in your way.

Try to imagine an aerial view of a 375 metre hold with a strong cross wind, a large lake right in the middle of the fairway with trees around it, with two bunkers infront of the green and two bunkers in front of the lake. (And this is a PAR 4 hole!)

The rules of golf apply in almost all instances. When you hit a ball into the lake there is a realistic splash and the computer gives you another ball. You are also penalised. It is also possible to lose your ball out-of-bounds by hitting it off the screen. This results in a penalty and you have to start again.

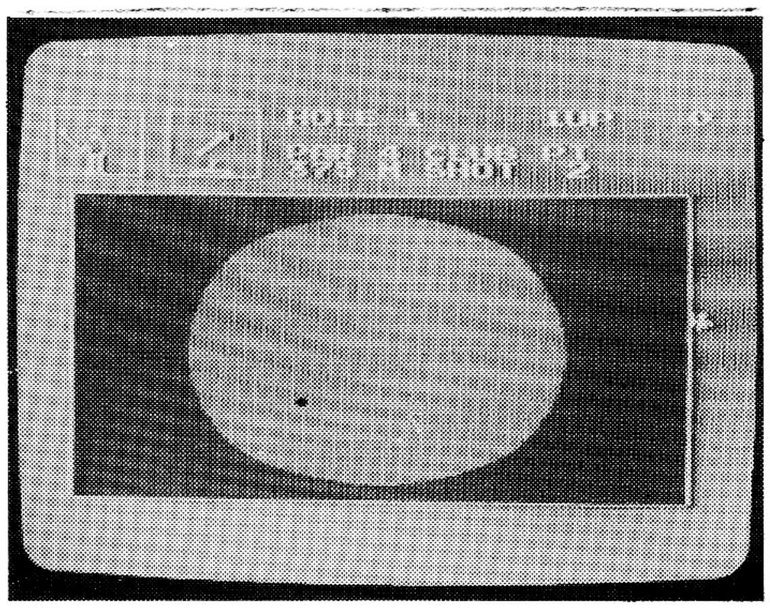
When you finally reach the green the computer automatically gives you your putter. The green is magnified so that it



takes up the entire screen. You now have to deal with the slope of the green which may be in any one of eight directions. This makes putting a real art as every time you hit the ball you have to careful to take into account the slope of the green. (fortunately this does not change with every shot).

When you have sunk the ball in the cup the computer moves onto the next hole automatically. When you have finished all nine holes the computer displays your score card so that you can see which holes you need to practice.

This cartridge would rate as one of the most realistic golf games there are available. You do not have to be a golfer to enjoy it. So if it's raining next time you want to play golf . . . it's time you played Sega Golf. All the thrills of golf without even leaving the house.



# ADDENDUM

We wish to inform you of alterations which can be made to the Sega File System Program, which will ensure more efficient operation when handling large numbers of files. This may have already been done in more recent copies.

The changes are as follows:-

```
150
FORDE=1TO250:NEXTDE:GOTO20

1005
DIMD(N+1),A$(N+1),B$(N+1,3),F(
N+1)

8190 GOTO8165
```

You may have also noticed that the Sprite Editor will not invert horozontally correctly.

If you wish to get your program working correctly, please change line 1370.

```
1370 ONI$1GOSUB1470,1480,1490,
1500,1490,1500,1470,1510,1480,
1500,1520,1500,1520,1510,1470
```

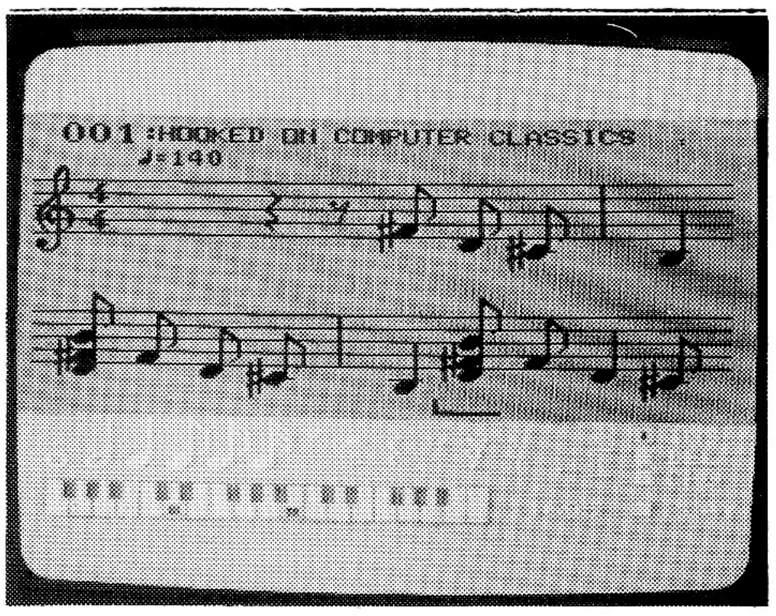
You may be aware that the last program listed on the Music Editor instruction sheet does not work, due to a typing error.

We suggest you make the following alterations to get this program running:-

```
20 READ CH, T, VOLUME, DELAY

30 SOUND CH, T, VOLUME: FOR D5O
TO DELAY:NEXT D
```

# Music Cartridge Demonstration

**Program by**
**Philip Bachler**

Have you ever heard a computer play The Fight of the Bumble Bee or The Entertainer?

This cassette contains ten pieces of music which demonstrate just what the Music Editor Cartridge is cabable of producing. All ten pieces of music can be stored in the music cartridges memory at once. The computer will play them repeatedly in any order as often as you like. The cassette has the following selection:

Hooked on Computer Classics
The Entertainer
Hark! The Herald Angel Sings
Beethoven's Minuet in G
Sailor's Hornpipe
From Russia with love
Yume-No-Tochyu
Nocture
Rydeen
Beautiful Dreamer

# SPRITES

Think of a Sprite as a piece of transparent film over your television screen. On this film you may draw a shape. Your Sega SC3000 has 32 Sprites, thus, you have 32 pieces of film which you may move independantly and draw on.

Each individual Sprite is made up of a group of 64 dots (8 by 8 block). You may turn "on" and "off" anyone of these dots. If a dot is turned "on" it will show up as part of a pattern, however, if the dot is turned "off" the background will show through. This allows you to move, say a Sprite of a car across a house drawn in the background and the background will show through the holes in the car. Sprites will also pass over the top of one another. This is called Sprite Priority, in other words, you can get a Sprite of a man to pass in front of a Sprite of a house, which you can also get the same Sprite of that man to pass behind the Sprite of say a tree.

Sprites are very useful in any program using graphics for games and education as you can animate your display making the program more lively and interesting and a lot more fun. All action games use Sprites for men, ships, bullets and targets etc.

## SPRITE CREATION

To create a Sprite we have to give the computer the following information:

1. What the Sprite is to look like.
2. Its position on the screen.
3. Its colour.
4. Its priority.

To tell the computer what the 8 by 8 block of dots (Sprites) looks like we use Hexidecimal Notation. This is a base 16 number system. What this means is that instead of 10 digits (0-9) like we have with the decimal number system, we have 16 digits. As there are not 16 digits we incorporate some letters as well. These are ABCDE and F. So Hexidecimal uses numbers 0-9, letters A-F. Each letter or number of hexidecimal corresponds to a group of dots or pixels.

Notice that the Binary Code is exactly the same as the graphic groups. Hexidecimal Notation is a shorthand method of Binary Code.

Now that we know how to define a group of 4 pixels we use the same ideas to define a Sprite to 64 dots or pixels. To tell the computer what a Sprite looks like we must first know ourselves, so it is easier to draw it on a piece of paper first.

Table 1

| Decimal (Base 10) | Hexidecimal (Base 16) | Binary Code |
|---|---|---|
| Ø | Ø | ØØØØ |
| 1 | 1 | ØØØ1 |
| 2 | 2 | ØØ1Ø |
| 3 | 3 | ØØ11 |
| 4 | 4 | Ø1ØØ |
| 5 | 5 | Ø1Ø1 |
| 6 | 6 | Ø11Ø |
| 7 | 7 | Ø111 |
| 8 | 8 | 1ØØØ |
| 9 | 9 | 1ØØ1 |
| 10 | A | 1Ø1Ø |
| 11 | B | 1Ø11 |
| 12 | C | 11ØØ |
| 13 | D | 11Ø1 |
| 14 | E | 111Ø |
| 15 | F | 1111 |

Ø = Pixel "Off" or unfilled
1 = Pixel "On" or filled

Shade in the dots you want turned "on" to form your Sprites (try using graph paper as it is a lot easier and quicket). Divide the 8 by 8 block down the middle.

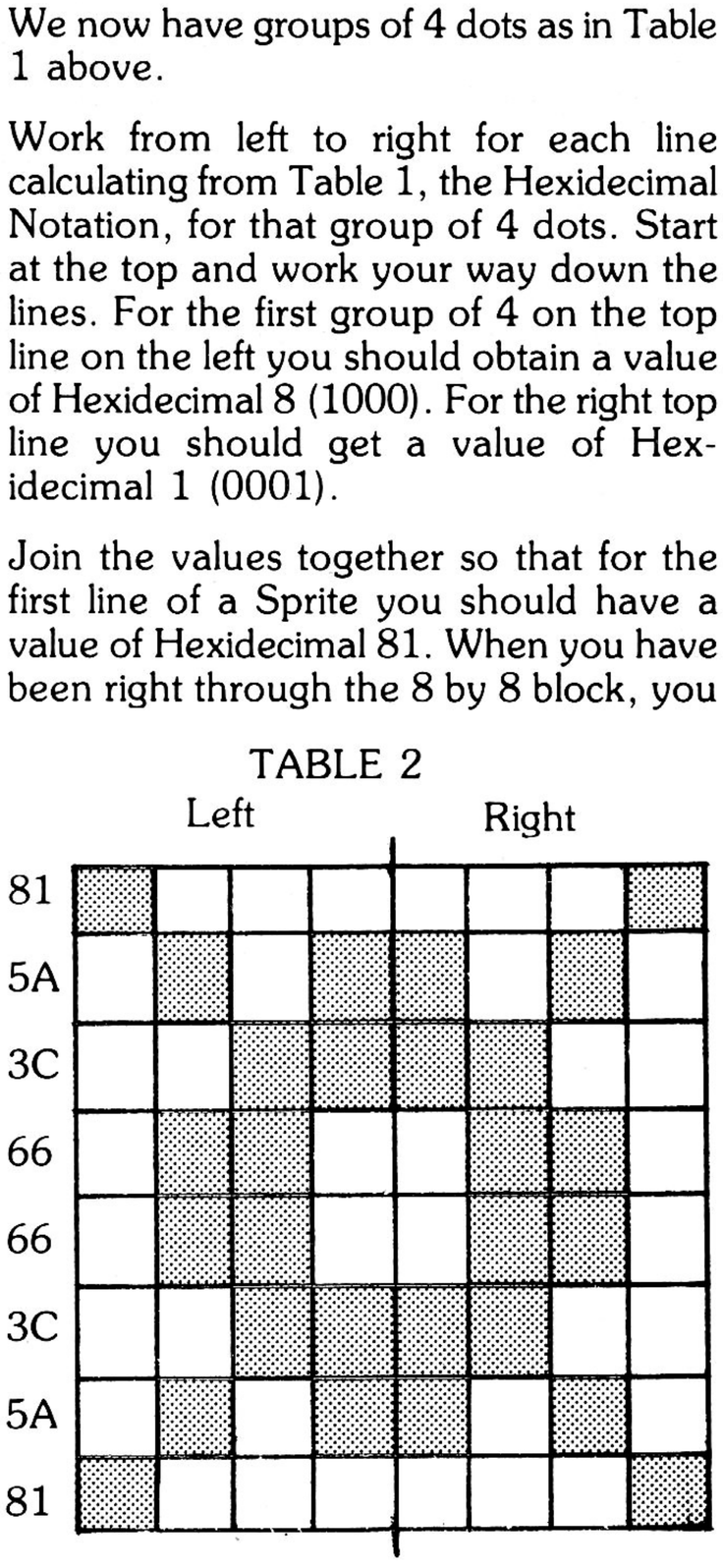
**See Table 2**

We now have groups of 4 dots as in Table 1 above.

Work from left to right for each line calculating from Table 1, the Hexidecimal Notation, for that group of 4 dots. Start at the top and work your way down the lines. For the first group of 4 on the top line on the left you should obtain a value of Hexidecimal 8 (1000). For the right top line you should get a value of Hexidecimal 1 (0001).

Join the values together so that for the first line of a Sprite you should have a value of Hexidecimal 81. When you have been right through the 8 by 8 block, you should have 16 Hexidecimal numbers. This is the information you give the computer to form a Sprite, that is 815A3C66663C5A81.

TABLE 2




We cannot just give the computer the Hexidecimal information. We have to use a pattern statement with the following format.

Pattern S#n, "Hexidecimal information."

In this case pattern S#O, "815A3C66663C5A81" the number n tells the computer which location in its memory to store the Hexidecimal Notation to form the Sprite. This number n must be between 0-225. We have now defined the Sprite to the computer. All it needs now is a colour, position and a priority. To do this we use a SPRITE statement with the following format.

SPRITE priority, (X co-ordinate, Y co-ordinate), n, colour.

Priority: This number labels Sprite and must be between 0-31. No two different Sprites should have the same Sprite number. The Sprite with a priority of 0 will be most prominant and will pass in front of all other Sprites, whereas a Sprite with another priority of 31 will be the least prominant and will pass behind all other Sprites, in other words, if two Sprites cross or "collide", the Sprite with the lowest priority number will pass in front of the other.

(X,Y): This tells the computer where to put the Sprite on the graphic screen. The X co-ordinate (across) must be between

0-255, whereas the Y co-ordinate (down) must be between 0-191.

This allows you to move the Sprite around the screen by changing the X and or Y co-ordinates (see examples at end).

N: This is the memory location where information for what the Sprite looks like is stored. This must be a number between 0-255.

Colour: This is the colour of the Sprite and must be a number between 0-15. Please see page 100 of the Level III manual for the colour table.

Now we can put the Sprite on the screen using the following program.

**Program 2**

```
10 SCREEN 2,2:CLS
20 PATTERN S#0,"815A3C66663C5A81"
30 SPRITE 1,(100,100),0,6
40 GOTO 40
```

Line 10: This sends the computer to the graphic screen (screen 2,2) and clears the screen (CLS).

Line 20: This fills the memory location 0 with the Hexidecimal information to create our Sprite.

Line 30: This positions the Sprite number one (the second most prominant Sprite) at position 100,100 (100 along, 100 down). It accesses memory location 0 for the Hexidecimal information to define the Sprite. It has a colour of 6 which is a Dark Red.

Line 40: This keeps the Sprite on the screen.

Now we want to move the Sprite so we change the X or Y or both co-ordinates with some sort of loop. The following example uses a For - Next loop.

```
10 SCREEN 2,2:CLS
20 PATTERN S#0,"815A3C66663C5A81"
30 FOR X=1 TO 255
40 SPRITE 1,(X,100),0,4
50 NEXT X
60 GOTO 30
```

Now change lines 30 and 40 to read:
Line 30: For x = 0 to 191.
Line 40: Sprite 1, (x,x),0,12.
Now try change Line 30 to read:
Line 30: For x = 0 - 191 Step 2.
The Sprite is now moving twice as fast.
Listed below are some further examples of what can be achieved with Sprites.

```
10 SCREEN 2,2:CLS
20 PATTERN S#0,"FFFFFFFF00000000"
30 PATTERN S#1,"00000000FFFFFFFF"
40 FOR X=1 TO 255 STEP 2
50 SPRITE 4,(X,100),0,6
60 FOR DE=1 TO 20:NEXT DE
70 SPRITE 4,(X+1,100),1,6
80 FOR DE=1 TO 20:NEXT DE
90 NEXT
100 GOTO 40
```

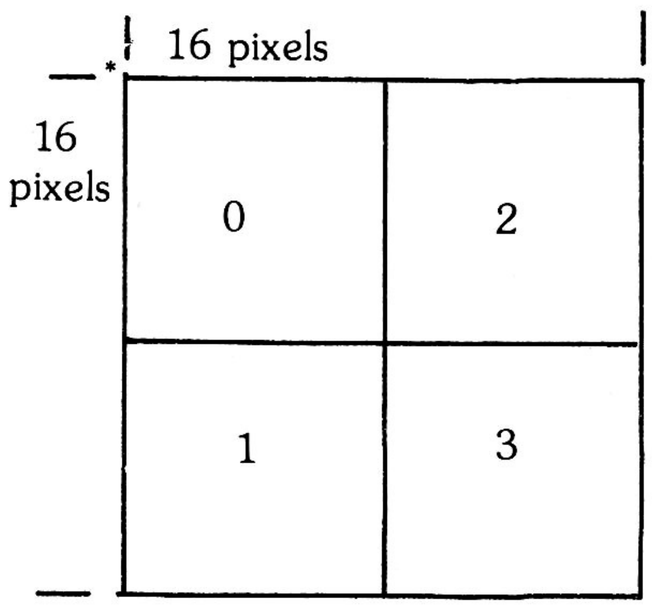
```
10 SCREEN 2,2:CLS
20 PATTERN S#0,"3C7EFFFFFFFF7E3C"
30 MAG0
40 X=0:Y=0:I=1
50 IF X<I THEN XX=I
60 IF X>247-I THEN XX=-I
70 IF Y<I THEN YY=I
80 IF Y>183-I THEN YY=-I
90 SPRITE 1,(X,Y),0,6
100 X=X+XX:Y=Y+YY
110 GOTO 50
```

```
10 SCREEN 2,2:CLS
20 C=1:COLOR 14,15,(0,0)-(255,191),14
30 FOR I=0 TO 255 STEP 16
40 FOR J=0 TO 191 STEP 16
50 IF C MOD 2=1 THEN COLOR ,14,(I,J)-(
I+15,J+15)
60 C=C+1
70 NEXT J:C=C+1
80 NEXT I
90 FOR I=1 TO 200 :NEXT I
100 COLOR 1,15
110 PRINTCHR$(17):PRINT"   SPRITE  TES
T "
120 PATTERN S#0,"FF999999999999FF"
130 PATTERN S#1,"FF999999999999FF"
140 PATTERN S#2,"FF999999999999FF"
150 PATTERN S#3,"FF999999999999FF"
160 PATTERN S#4,"0103070F1F3F7FFF"
170 PATTERN S#5,"D5AAD5AAD5AAD5AA"
180 PATTERN S#6,"80C0E0F0F8FCFEFF"
190 PATTERN S#7,"55AB55AB55AB55AB"
200 PATTERN S#8,"1818FF18FF181818"
210 PATTERN S#9,"1818181818181818"
220 PATTERN S#10,"1818FF18FF181818"
230 PATTERN S#11,"1818181818181818"
240 PATTERN S#12,"0000FF7F4343437F"
250 PATTERN S#14,"0000FCF0301010F0"
260 MAG 3
270 X=1:Y=100
280 SPRITE0,(60,Y),0,5
290 SPRITE8,(100,Y),4,8
300 SPRITE2,(140,Y),8,3
310 SPRITE5,(X,Y),12,1
320 PATTERN S#13,"FFFFFFC399BD2418"
330 PATTERN S#15,"FEFFFFC399B5B518"
340 PATTERN S#13,"FFFFFFC399AD2C18"
350 PATTERN S#15,"FEFFFFC399BDA518"
360 PATTERN S#13,"FFFFFFC399A53C18"
370 PATTERN S#15,"FEFFFFC399ADAD18"
380 PATTERN S#13,"FFFFFFC399B53418"
390 PATTERN S#15,"FEFFFFC399A5BD18"
400 T=T+1:IF T>=250 THEN 430
410 X=X+2
420 GOTO 310
430 GOTO 270
```

Now that we have constructed a sprite and moved it around the screen, we can now magnify it using the MAGnifying statement. There are four magnifications (0-3).

MAG 0 - This is the standard 8 x 8 pixel sprite that has been used in the examples up until now.

MAG 1 - This is a group sprite and is made up of four 8 x 8 pixel sprites, which are treated as one so you end up with a 16 x 16 pixel sprite.




* Access position

When you access a sprite with a magnification of 1, the number n should be a multiple of four. The computer automatically reads the next four memory locations for the hexidecimal information. This saves a lot of time when programming, as you do not have to move all four parts of the sprite individually.

When positioning a Mag 1 sprite, the X and Y, co-ordinates that you give the computer, are for the top left hand corner of the group of sprites.

```
10 SCREEN 2,2:CLS
20 PATTERN S#4,"FF80808080808080"
30 PATTERN S#5,"FFFFFFFFFFFFFFFF"
40 PATTERN S#6,"FFFFFFFFFFFFFFFF"
50 PATTERN S#7,"01010101010101FF"
60 MAG1
70 FOR X=0 TO 255
80 SPRITE 1,(X,100),4,12
90 NEXT X
100 GOTO 70
```

MAG 2 - This is exactly the same as Mag 0, except the sprite is twice the size (in other words, a 16 x 16 pixel block).

```
10 SCREEN 2,2:CLS:M=0
20 PATTERN S#9,"0103070F1F3F7FFF"
30 MAG M
40 FOR X=0 TO 255
50 SPRITE 1,(X,100),9,RND(1)*13+1
60 NEXT X
70 IF M=0 THEN M=2:GOTO 30
80 M=0:GOTO 30
```

MAG 3 - This is the same as Mag 1, so the sprite is twice the size, thus you have four 16 x 16 pixel sprites (a group sprite 32 x 32 pixels!)

# READER'S LETTERS

**Mr F.J. Wydur**
**406 West Coast Rd**
**Glen Eden**
**Auckland 7**

**16 August 1984**

**Mr Philip Kenyon**
**Sales Manager Computer Division**
**Grandstand Leisure Ltd**
**P.O. Box 2353**
**AUCKLAND**

Dear Sir,

In receipt of SEGA USERS CLUB invitation, thank you; Yes, I would be more than interested to join your club, therefore please find enclosed cheque for the subscription fees.

I would like to take this opportunity to express my pleasure at owning the Sega computer, being a parent with three young children, I have gone overboard to get them involved in computer work, as I believe that is where their future livelihood will be.

As a result of this belief, I purchased an Atari 400 well over a year ago, and had memory expanded out to 52K, plus also buying the Sega SC 3000.

I find the Sega a real pleasure to use, and feel it is well superior in both ease of use, and function, to the Atari.

More recently, my eldest son has been bringing an Apple two e home from High School, and having had use of the three mentioned machines find I prefer the Sega above the others; thank you for a fine product.

Yours sincerely
**F.J. Wydur**

# READER'S LETTERS

## DEAR EDITOR

Not having received the magazine as yet, I am not too sure who to send this too.

I recently purchased the Sega SP400 to go with my Sega 3000.

Included in the Users Manual is several demo programs. The one I have included won't run giving me a syntax error in line 20. True I PI does equal 3.142, but it appears my computer requires something more. With my limited experience I am not sure what. Can you help?

**W P Andersons**

```
10 DIM A(10,2)
20 PI=3.142 <-- Syntax error ?
30 REM ***PLOTTING ABILITY
40 LPRINT :LPRINT TAB(12);
50 LPRINT "PLOTTING ABILITY"
60 LPRINT CHR$(18);"LO"
70 LPRINT "M250,-180":LPRINT "1"
80 FOR I=0 TO 350 STEP 10
90 S=I/180*PI
100 X=SIN(S)*200.5
110 Y=COS(S)*200.5
120 X=INT(X):Y=INT(Y)
130 LPRINT "D";X;",";Y:LPRINT "H"
140 NEXT I
150 LPRINT "M0,-450"
160 LPRINT "I"
170 S=2*PI/11
180 FOR I=0 TO 10
190 A(I,1)=INT(SIN(I*S)*200.5)
200 A(I,2)=INT(COS(I*S)*200.5)
210 NEXT I
220 LPRINT "M";A(0,1);",";A(0,2)
230 C=2
240 FOR I=0 TO 4
250 K=0
260 C=C+1:IF C>3 THEN C=1
270 LPRINT "C";C:FOR WA=0 TO 1000
280 NEXT WA
290 FOR J=0 TO 10
300 K=K+I+1
310 IF K>10 THEN K=K-11:GOTO 310
320 LPRINT "D";A(K,1);",";A(K,2)
330 NEXT J
340 NEXT I
350 LPRINT "D";A(K,1);",";A(K,2)
360 LPRINT "M0,-200":LPRINT "C0"
370 LPRINT CHR$(17)
380 END
10 OUT &HDF,&H9B
20 COLOR 4,15
30 FOR DE=1 TO 100:PRINT DE;:NEXT DE
40 PRINT
50 OUT &HDF,&H9A
60 COLOR 6,15
70 FOR DE=1 TO 100:PRINT DE;:NEXT DE
80 GOTO 10
```

## EDITORS REPLY

The problems that you are having with the program which you enclosed, are very easy to rectify by carrying out either of the following:

1. Delete Line 20 altogether. The reason that you are getting a syntax error on this line is that the computer already knows what PI is.

2. You can change Line 20 so that it reads something like PA=3.142 and everywhere else in the program where PI appears (Line 90, Line 170).

This will remedy the situation as the computer has now defined a new variable as 3.142, instead of trying to redefine PI, which is not possible.

## DEAR EDITOR

A short time ago I completed my first game for the Sega. However until I received the first issue of the Sega Users Magazine and read the article concerning Sega's young programmers, I had not thought of sending it in to you.

The game is 7K and of course written in BASIC. The setting is in a western town and there are approximately 2K of Hi Res graphics written by 'data' which draws a saloon and a cactus. Has explained in the instructions you are a gunfighter on the left and you must shoot the other gunfighter opposing you on the right. You walk towards each other and on the count of fifteen steps, the word DRAW appears on the screen and this is the time to fire. If, however, you fire before time the computer will automatically shoot you, so this rules out holding down the key. If you are successful in shooting your opponent then there is a quick break and the computer shows you your score and time you've been playing, then you are returned to SCREEN 2. If you are unsuccessful the computer will show you your score, hi-score, time of play, and your rating: Pathetic Cretin — Marsahall. The game gets harder as play continues.

Since writing this program I have almost completed my second program, and have realized that in some cases there were better ways of writing GUNSLINGER. Unfortunately I have not yet purchased the SP-400 printer/plotter, but hope to in the near future, so I hope you can decipher my printing.

Please reply to this letter and tell me what you think of my program. If you feel it is good enough you may print it in the magazine. If you do not like it remember I am only 14.

**Glen Howard**

## EDITORS REPLY

Yes Glenn we did like your game, and we're sure a lot of others will too. Keep your eye on the post!! A thank you on the way.

## DEAR EDITOR

I am having some problems getting my Cassette Recorder to load and save programs. Do you have any suggestions as to how I could do this more successfully? Do I require a Sega Data Recorder?

Yours faithfully
**Mrs V Seaton**

## EDITOR'S REPLY

There are many reasons that may be causing problems saving and loading information on your Cassette Recorder. We suggest you try the following:

1. Clean the heads and the rubber pinch roller on your Cassette Recorder regularly with meths (or a head cleaner solution) and a cotton bud.
2. Use C10 or C20 computer tapes, as C30, C60 and C90 Audio tapes are too thin. They twist, stretch, and snap more easily. They are also longer and heavier, so more strain is placed on the Cassette Recorder motor.
3. Ensure that the heads on your Cassette Recorder are properly adjusted and are in alignment with the recorder tape.
4. Have the heads of the Cassette Recorder demagnetised.

Obviously with the Sega Data Recorder, you would have fewer problems if it was properly adjusted, as you would not have to worry about tone or volume settings.

## DEAR EDITOR

For one of my programs I would like to be able to turn "off" or disenable the keyboard. Is this possible and is so how?

**Mrs C R Riely**

## DEAR EDITOR

Can you please tell me how to stop a program being listed so pestering little people cannot muck about or zap my programs.

**G Sydenham**

## EDITORS REPLY

It is very easy to disenable the keyboard using the following line of programming:

OUT &HDF, &H9B

The only key that is still active is the RESET key which when pressed resets the computer and re-activates the keyboard. The program in the computers memory is not damaged.

The following line enables the keyboard again.

OUT &HDF, &H9B

Try the following program:

```
10 OUT &HDF,&H9B
20 COLOR 4,15
30 FOR DE=1 TO 100:PRINT DE;:NEXT DE
40 PRINT
50 OUT &HDF,&H9A
60 COLOR 6,15
70 FOR DE=1 TO 100:PRINT DE;:NEXT DE
80 GOTO 10
```

When the screen is blue (caused by Line 20) the keyboard is disenabled except for the reset key. This has been achieved by Line 10. You cannot break the program at this stage.

When the screen is red (see Line 60) you may break the program using the break key as they keyboard has been re-activated by Line 50.

This will not stop people listing your programs, but it will stop them breaking the program, changing some variable and continuing.

---

## DEAR EDITOR

I have noticed on my Sega computer, that the command RND(0) does not act in the manner described in the Manual. After a number of trials, I think that I can now be certain that PRINT RND(0) of will always produce .38502125073. Further, it appears that use of RND(0) will always reset RND(1) to the same sequence which commences .876463237.

Thus it is essential in any game to avoid use of RND(0) folllowed by RND(1), as the game will produce the same sequence every time. RND(-1) however, does restore an unpredictable result again.

Is this a fault on my machine, or is the manual incorrect?

Another peculiarity which occurs, is that sometimes on switch on, there is no display on the T.V., only a beat pattern and an unusual tone on sound. This is not an unlocked display, the computer is definitely not doing anything sensible and none of the keys have any effect. The only cure is to switch off and on again a little later. Although this generally seems to result in having 0 byte of memory available, this in turn seems to be best tackled by loading in a program which in time, is rejected by the Sega, but restores everything to normal.

The above only happens occasionally and I wonder whether perhaps one of the oscillators may be the problem.

Any suggestions?

Yours faithfully
**K O Surridge**

| | |
|---|---|
| .64688917744 | .46513709949 |
| .71388780445 | .15372732329 |
| .81079424393 | .46068220205 |
| .53310458173 | .26798422764 |
| .75648824295 | .77907921069 |
| .19932479303 | .32714072506 |
| .073617552162 | .56365732247 |
| .69002036461 | .10900663353 |
| .72041200588 | .51344517906 |
| .049630839541 | .20415222379 |
| .26798422764 | .43012942978 |
| .46081912165 | .23012051918 |
| .15372732329 | .98938698166 |
| .38502125073 | .69057544491 |
| .46405018929 | .71553970099 |
| .13855443831 | .64080098809 |
| .45115685401 | .83056602439 |
| .039439217605 | .54345164444 |
| .91888534769 | .54343457393 |
| .53538318662 | .010517160714 |
| .14144544255 | .39038611188 |
| .26798422764 | .43012942978 |
| .38502125073 | |
| .034190055573 | .87327778367 |
| .69239679743 | .49983079664 |
| .52934124499 | .46029091285 |
| .48943935935 | .85472087505 |
| .65625102461 | .14697513926 |
| .80435003423 | .18944853351 |
| .41867757949 | .22707945546 |
| .049630839541 | .38307029594 |
| .052865446886 | .034190055573 |
| .29347754257 | .33253429293 |
| .27733746577 | .80144559865 |
| .14697513926 | .34690150686 |
| .60628553427 | .55653931927 |
| .32714072506 | .13936307222 |
| .56365732247 | .049630839541 |
| .13124431205 | .19465468623 |
| .46029091285 | .13855443831 |
| .90423374312 | .58377709639 |
| .21222562832 | .24002795088 |
| .9894637309 | .38031639431 |
| .22707945546 | .88011494586 |
| .54017096798 | .99810486624 |
| .38502125073 | |
| .69057544491 | .80351608507 |
| .81079424393 | .19863101707 |
| .13124431205 | .93901830129 |
| .55938452367 | .081452292467 |
| .68280348443 | .19932479303 |
| .039439217605 | .64688917744 |
| .066988727126 | .54343457393 |
| .12209035271 | .72041200588 |
| .69057544491 | .27733746577 |
| .71553970099 | .07179971957 |
| .19082427309 | |

```
10 FOR I=1 TO 22
20 LPRINT RND(-1),:LPRINT RND(-1)
30 NEXT I
40 LPRINT RND(0)
50 GOTO 10
```

## EDITOR'S REPLY

The explanation of the RND function on page 77 of the Level 3 manual is somewhat misleading. For a start, RND does not produce a random number, but a psuedo random number. The computer has a sequence of numbers between 0 and 1 programmed into the Basic ROM that is reads each time a random number is required.

RND(0)

This produces the first psuedo random number .38502125073 (the one at the top of the last program in the Basic random). This is how you are getting a repeating list of random numbers. It is as you have correctly stated, essential in any game to avoid the use of RND(0), followed by RND(1), as the game will produce the same sequence.

RND:1

This reads the next random number in the list in the computer's memory.

RND-1:

This reads a random number for any position in the list of random numbers. This gives you a 'truly' random number.

Try the program listed below instead of the one you have sent in. Notice it does not repeat itself. Try using RND-1 in future, instead of RND1.

The problems you are having with your display are probably not due to a computer fault. These may be caused by your cartridge not being plugged in properly. To remedy this, reinsert the cartridge (always switch your machine off when inserting cartridges. A dirty connection on the cartridge may also be causing the problem. Try cleaning the metal contacts on the cartridge with Iso Propyl Alcohol, or Methylated Spirits.

---

## DEAR EDITOR

I am writing to ask your opinion on three things.

1. Is it possible to program in another language other than Basic, on the Sega?
2. Would it be possible to explain to me how to use the sprite editor? I do not quite understand the code used, which is a shame seeing as how the Sega has such good sprite facilities.
3. The Sega produced a very strange fault when I loaded up 'Sea Battle' (out of your Magazine), off my Sega Data Recorder, everything in the program's display was in the Mag 3 mode.

I hope you can help me. Great magazine, keep it up.

Yours faithfully

**M Smith**

## EDITOR'S REPLY

Currently there are Logo and Forty languages being written for the Sega. Unfortunately these will not be available until after Christmas.

It is also possible to program the low level assembly language (machine code), directly into the Sega.

For more information on sprites, please see the section of this Magazine detailing their creation and manipulation.

The problems you had with the Sea Battle were probably caused by the program previously in the computer's memory. If the computer is set in the Mag 3 mode and with double writing, (CHR$(17)) it will remain like this until Reset is pressed, or the program removed it from these modes.

## DEAR EDITOR

I would like to know when an Acoustic Modem and software will be available for use with the Disk Drive.

**T Shadbolt**

## EDITOR'S REPLY

Sega Japan is producing an Acoustic Modem for the New Zealand telephone system. (We do not have any firm dates to when this will be available but it will be sometime next year), however if you have a Disk Drive there is an RS232C port to which you can interface a third party modem.

Most of the cassette based softward sold now will run on the Disk Drive. Disk handling software is built into the basic of the machine.

# Hardware Information

1. Hardware information.
2. Keyboard & Joystick Matrix.
3. Work Ram & PIO information.

The following information will be necessary to develop machine language program on cartridge board.

1. cpu : 8oA CLOCK 3.58MHz

2. WARK RAM : ADDR. CØØØ c7FFH.

3. CRT (TI9918 or 9928) : port No. BEH as Data Register
   port No. BEH as Command Register

4. PSG (TI76489) : port No. 7FH

5. PIO (intel 8255) : port No. DCH as accessing to PIO port PA
   port No. DDH as accessing to Pio port PB
   port No. DEH as accessing to Pio port PC
   port No. DEH as chip control port No.

6. System Interrupts init. DATAØ92H
   ADDR. ØØ38H INT, cpu imitialized INT. MODE 1 (IM1)
   ADDR. ØØ66H NHI; RESET SW. is connected directly.

7. Key HATRIX

Please refer to diagram ↓

### WORK RAM AND PIO INFORMATION

WORK RAM
SC-3ØØØ CØØØH C7FFH

PIO (8255)
PORT C

| | | |
|---|---|---|
| PCØ | KEY AND JOYSTIC | |
| PC2 | | |
| PC3 | NO USE | |
| PC4 | CASSETTE OUT | |
| PC5 | DATA | |
| PC6 | RESET | PRINTER |
| PC7 | FEED | |

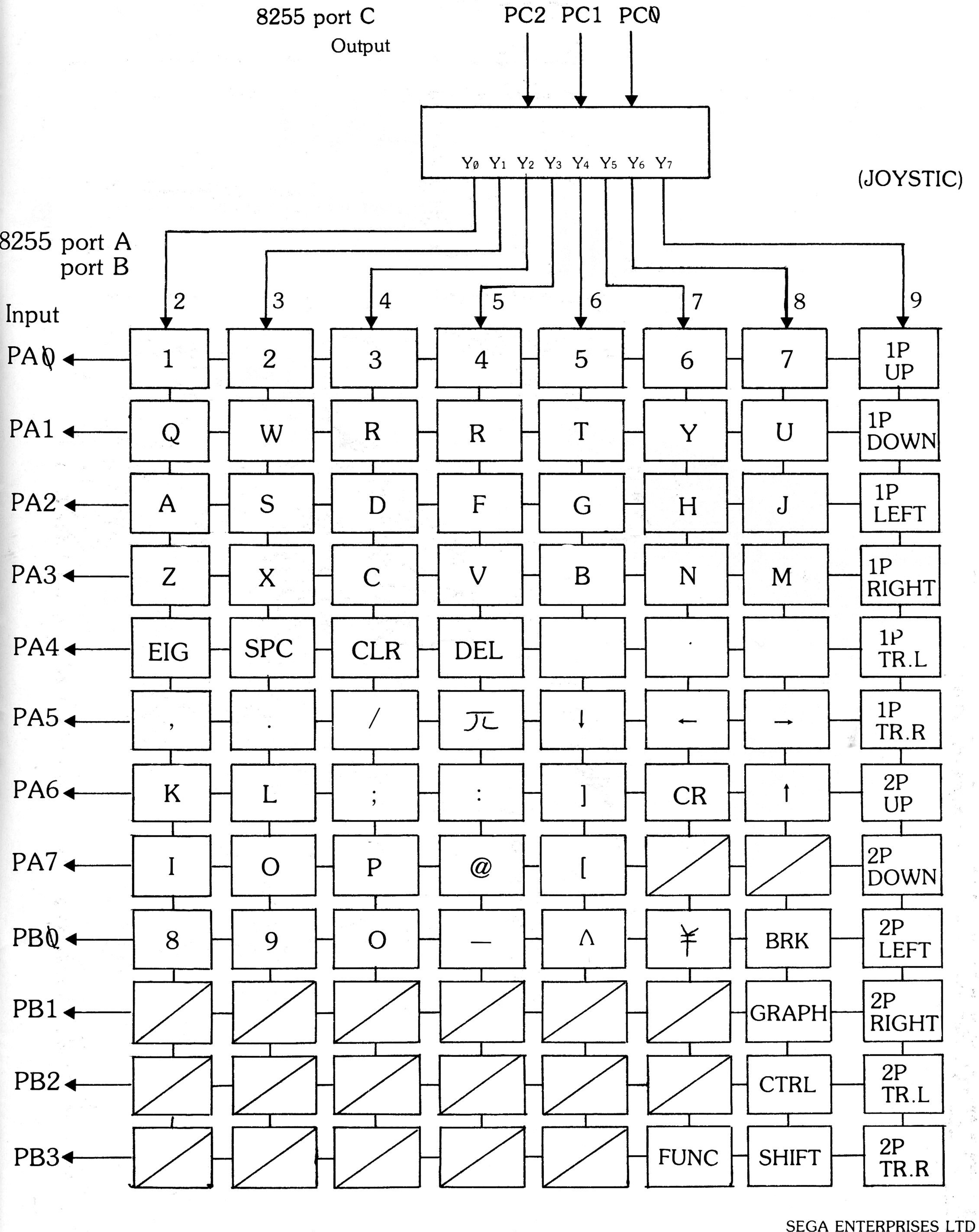
PORT B

| | | |
|---|---|---|
| PCØ | | |
| PB1 | KEY AND JOYSTIC | |
| PB2 | | |
| PB3 | | |
| PB4 | NO USE (CON) | |
| PB5 | FAULT | PRINTER |
| PB6 | BUSY | |
| PB7 | CASSETTE IN | |

### CARTRIDGE CONNECTOR ASSIGNMENT

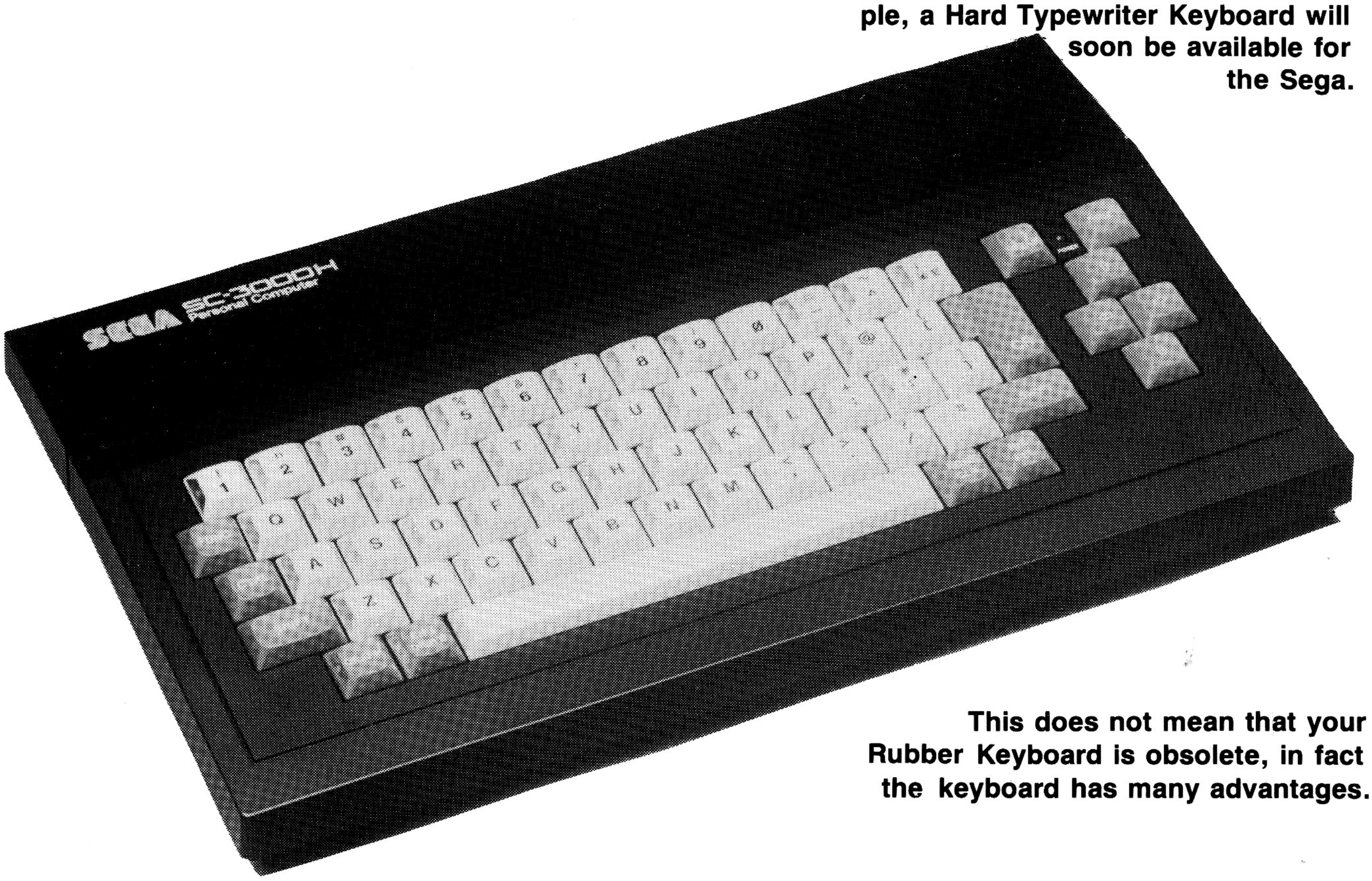
| SOLDERING SIDE | | PARTS SIDE | | | |
|---|---|---|---|---|---|
| AD | 1 | 1 | VCC | 5V | |
| A1 | 2 | 2 | VCC | 5V | |
| A2 | 3 | 3 | | | DSRAM |
| A3 | 4 | 4 | | | CEROM2 |
| A4 | 5 | 5 | RD | | |
| A5 | 6 | 6 | WR | | |
| A6 | 7 | 7 | | (OPEN) | I/OR |
| A7 | 8 | 8 | | (OPEN) | I/OW |
| A8 | 9 | 9 | | (REFR) | |
| A9 | 10 | 10 | MREO | | |
| A10 | 11 | 11 | | | CON |
| A11 | 12 | 12 | | (OPEN) | RASI |
| A12 | 13 | 13 | | (OPEN) | CASI |
| A13 | 14 | 14 | | (OPEN) | RAMA7 |
| Do | 15 | 15 | | (OPEN) | RAS2 |
| D1 | 16 | 16 | | (OPEN) | CAS2 |
| D2 | 17 | 17 | | (OPEN) | MUX |
| D3 | 18 | 18 | A14 | | |
| D4 | 19 | 19 | A15 | | |
| D5 | 20 | 20 | | OPEN | |
| D6 | 21 | 21 | GND | | |
| D7 | 22 | 22 | GND | | |

# 2 KEY BOARD AND JOYSTICK MATRIX



# SEGA is now even harder to beat

**For some funny reason there is a section of the population who look at a rubber keyboard and immediately start convulsing or run screaming from the room. Fortunately help is on the way for such people, a Hard Typewriter Keyboard will soon be available for the Sega.**




**This does not mean that your Rubber Keyboard is obsolete, in fact the keyboard has many advantages.**

1. The keys have carbon contacts of metal ones, as in most typewriter keyboards. These carbon contacts cannot corrode, become coated in oxide layer, cracked or wear out, thus your keyboard has a very long life expectancy.
2. The keyboard is lighter and so more portable.
3. There are no moving parts in the key pad so it is less likely to break or malfunction.
4. The main keyboard is made up of just one piece of rubber so it is semi-water proof and dust proof.

In October a Hard Keyboard Conversion Kit will be on your dealers shelves, which will sure to be of tremendous interest to all those serious users, or touch typists who at present find that there fingers moving somewhat faster than the speed at which the rubber keys allow.

A hard keyboard has currently the same layout as a soft one so if you have mastered using the rubber keyboard, you will have no problems using the new one.

The new key pad also has a 4 length space bar allowing touch typists to operate it more easily.

The keys are molded to prevent your fingers from slipping and will have all the notation embossed on them. Each key has a very positive action when hit, making the typing, for any reason, quick, easy and enjoyable.

SC3000H (The hard keyboard computer) is an absolute dream to use, and is assured of making the computer more enjoyable to use for anyone, whether you are a typist or not.

# COMPUTER INPUT

WIN A HOME COMPUTER

magazine each month carries competitions and regular columns bringing news of whats happening on the New Zealand computer scene with programs and articles specifically written for the Sega Computer by our regular columnists every issue.
Make sure you've got the latest for your Sega

## JOIN THE TEAM!

**ERIC McCALL**
**JACK NOBLE**
**ASHLEY NOBLE**
**JOEL NIELSEN**
**PHILLIPPA CHAPMAN**
**OLWEN WILLIAMS**
**BRIAN BROWN**
**TOM CHERRY**
**MARTIN HALL**
**FAYE HALL**
**SHAYNE BURBERY**
**BRUCE JOHNSTON**
**PETER FINLAY**

Published by –
NOMAC PUBLISHING LTD

SEND ME A 12 MONTH SUBSCRIPTION TO **COMPUTER INPUT MAGAZINE AT THE SPECIAL RATE OF $13.00 (11 ISSUES)**

NAME: .....

ADDRESS: .....

.....

..... PH .....

Computer Model..... I Own / Would Like

COMPUTER INPUT

**SUBSCRIBE TODAY**
**ONLY $13**
**NO STAMP REQUIRED**

I enclose payment: ✓
☐ Visa ☐ Bankcard

Cardholder signature.....

Date card expires ..... ..... .....

Cheque ☐ Cheque No.....

Postal Order ☐ Postal note ☐

**FREE POST NO. 671**
**Nomac Publishing Ltd,**
**P.O. Box 39-278**
**Auckland West**

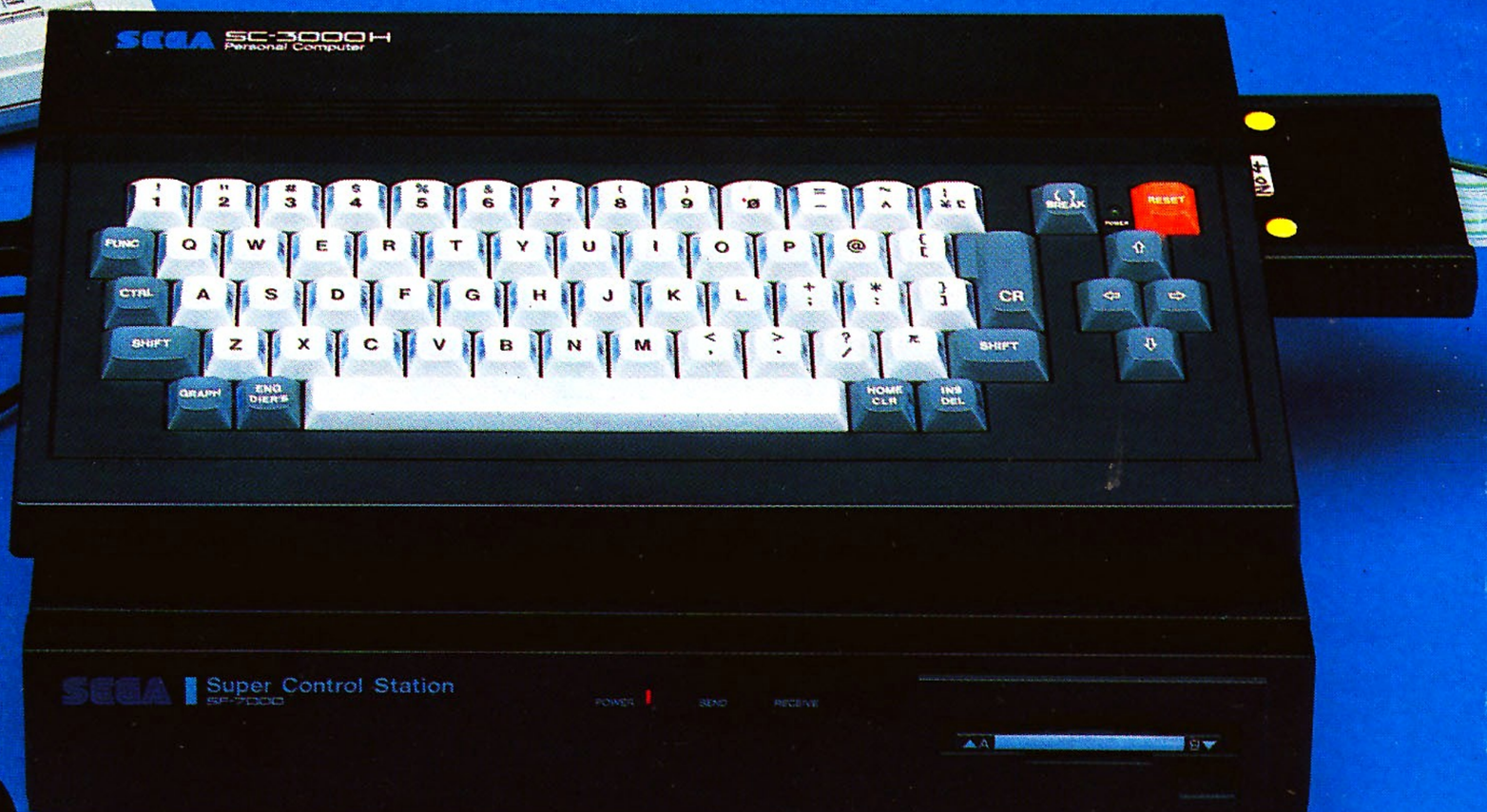
**Or Phone us AK: 496-943**

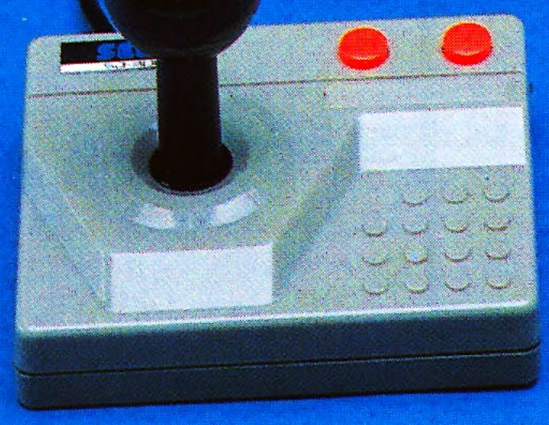
# The Hard Word from Sega

**SEGA'S new hard keyboard and new hard disk drive, make Sega even harder to beat.**

**The very latest industry standard 3'' diskettes store a massive 328K of information each desk, and operate at speeds far in excess of their older 5½ inch counterparts.**

**Sega's super control station will also enable you to connect to many other attachments such as business printers electronic typewriters even telephone moderns, through it's two additional 1/0 ports, for RS232C and parrallel interfaces.**





Sega now boasts not only the most sophisticated and easy to learn basic language, the most incredible graphic capabilities, a sensational range of cartridge and cassette games, superb education and applications software written by and for New Zealanders, but also a very sophisticated, fast, small business system, with the state of the art Sega control station.
Feel the difference, see the difference, compare the difference, and you too can enjoy the difference of Sega superiority.
**Call at any Sega stockist for a demonstration or contact Grandstand Leisure at: Box 2353 AUCKLAND**

GRANDSTAND
SEGA®